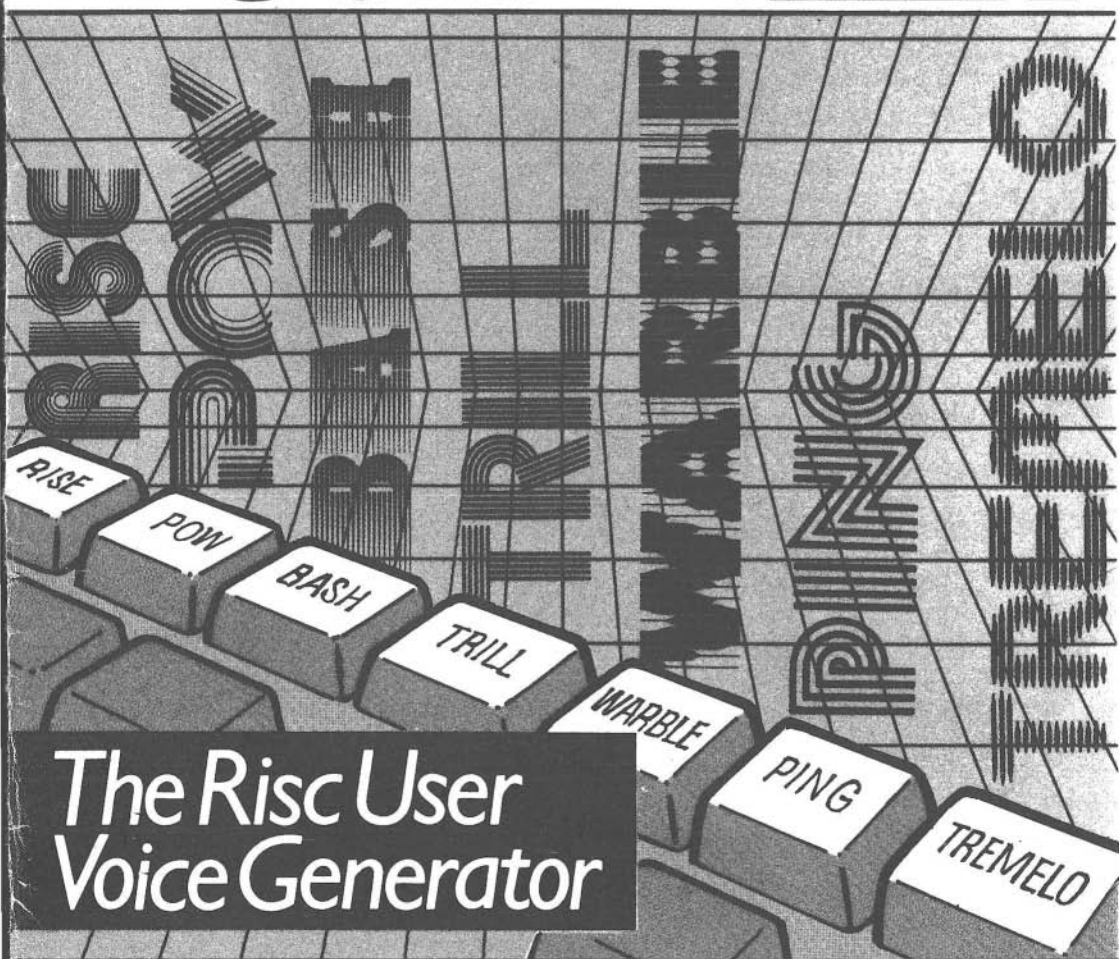


Volume 1
Issue 4

March 1988

Price £1.20

RISC USER



**The Risc User
Voice Generator**

**THE MAGAZINE AND SUPPORT GROUP
EXCLUSIVELY FOR USERS OF THE ARCHIMEDES**

RISC USER

CONTENTS

Editorial	3
News and Comment	4
The RISC User Voice Generator	5
How to get new sounds from your Archimedes	
Archimedes 440 Review	8
What does the latest Archimedes, the A440, have to offer	
Archimedes Visuals	10
More impressive visual effects.	
1st Word Plus	12
A new and powerful word processor from Acorn	
A Full Screen Pixel Editor	14
Create and edit full screen images	
Logical RAM Tracer	19
Trace your logical RAM with this utility	
Writing a Relocatable Module	20
How to write your own relocatable modules	
Getting the Archimedes On-line	23
A survey of comms packages for the Archimedes	
At the End of the Rainbow	24
How to re-define the palette in 256 colour modes	
A Mouse Pointer Designer	29
Design your own mouse pointer with this easy to use program	
The Archimedes I/O Podule	31
We review Acorn's first podule	
Hints & Tips	34
Some more useful hints from our experts	

RISC User is published by BEEBUG Ltd.

Co-Editors: Mike Williams, Dr Lee Calcraft

Assistant Editor: Kristina Lucas

Production Assistant: Yolanda Turuelo

Technical Assistant: Lance Allison

Subscriptions: Mandy Mileham

All rights reserved. No part of this publication may be reproduced without prior written permission of the Publisher. The Publisher cannot accept any responsibility whatsoever for errors in articles, programs, or advertisements published. The opinions expressed on the pages of this journal are those of the authors and do not necessarily represent those of the Publisher, BEEBUG Limited.

BEEBUG, Dolphin Place, Holywell Hill, St.Albans, Herts
AL1 1EX. Tel. St.Albans (0727) 40303

BEEBUG Ltd (c) 1988



The Archimedes Magazine and Support Group.

EDITORIAL

ARTHUR UPGRADES

If your Archimedes was supplied with a 0.2 or 0.3 operating system, you should by now have received your Arthur 1.2 upgrade. If you have not, and you sent off the machine registration form at least a month ago, then you should write to Acorn at the address below giving your full name and address, and the serial number of your machine. Because all readers should now be using 1.2, we have not made any attempt in this issue to check that our programs will run correctly on earlier operating systems, though as usual we have included a note where any program requires special machine settings.

Most readers will by now also have received the ArcWriter word processor which Acorn are mailing free of charge to all currently registered users. If you have not received yours by the time that you read this, you should hang on for a further 10 days or so, and then write to Acorn at the address below, again giving full name, address and machine serial number.
Acorn Computers Customer Services, Fulbourn Road, Cherry Hinton, Cambridge CB1 4JN.

SERIAL BUG

As you may be aware, the serial port on the Archimedes still does not function correctly, even with the O.S. 1.2 upgrade. In fact on O.S. 1.2 the problems are even worse, as Ian Burley reports in our Comms article this month. In their defence Acorn argue that the problems all arise from bugs in the 6551 serial chip which they are using. You will be pleased to hear though, that they have come up with a "fix" which solves the most obvious of the problems. This comes in the form of a relocatable module, and Acorn will be supplying this on future Welcome discs, and will be offering an upgrade to present users. In the mean time we are providing an early version of the fix on this month's magazine disc (it is too large to go into the magazine itself). This month's disc also contains a data file for the Welcome Music Editor which plays part of Grieg's Peer Gynt Suite No.1.

PROGRAMMING IN "C"

RISC User's sister magazine BEEBUG is running a series of 4 or 5 articles on programming in "C". The series, which offers a down-to-earth approach to programmers familiar with Basic, begins in the March issue of that magazine. Although space does not permit us to reproduce these articles in RISC User, we feel that they will be of interest to many of our readers. We are therefore offering to supply copies of the articles to RISC User members for a small supplementary payment. Please see the leaflet accompanying this issue for further details. Starting next month we shall also be including the "C" articles on the RISC User magazine disc.

NewsNewsNewsNewsNews

A round-up of the latest news and comment in the Archimedes world compiled by Mike Williams.

ACORN NEWS

As stated in the editorial Acorn expects to have shipped out all replacement 1.2 ROM sets by the end of February. Your copy of ArcWriter should arrive about a week after the 1.2 ROMs. Acorn has said that it will continue to supply ArcWriter free to all new Archimedes owners up to the end of March this year. In its usual fashion, Acorn has managed to send two ROM sets to some customers, and has followed this up with a letter asking for the return of one of the sets. Unfortunately, this letter has also been received by some unhappy Archimedes users still waiting for their first replacement ROM set!

1st Word Plus (see the review in this issue) is forecast for release at the end of March, though the price still has to be fixed, and Kermit, the public domain communications software, is due out at about the same time for £56.35 (to cover documentation and packaging says Acorn). Version 1.09 of the PC Emulator offering greater compatibility and some increase in speed has now been released. Existing users may upgrade to the latest version through BEEBUG by returning the original PC Emulator disc (NOT the MS-DOS disc) with a handling fee of £10 which includes postage.

The Acorn ROM podule is now in stock, but the hard disc upgrade for the 300 series is not expected until April.

LOGO FOR THE ARCHIMEDES

Logotron, originators of the best selling Logo for the BBC series has announced an Archimedes version operating at up to ten times the speed of the original and with a very much larger workspace. Graphics using Logo now offers the possibility of 1024 by 960 pixel resolution with up to 256 colours in certain modes, and the option of a 132 column editor. Archimedes Logo costs £60.00 from Logotron Ltd, Dales Brewery, Gwydir Street, Cambridge CB1 2LJ or telephone (0223) 323656.

ARCHIMEDES GETS THE PINEAPPLE TREATMENT

Pineapple Software, well known for their graphics software for the BBC micro has announced that Diagram II is now available to run under the 6502 emulator on the Archimedes. Diagram II is particularly suited to all kinds of schematic diagrams made up from a set of standard

symbols (like circuit diagrams for example), and has been very favourably reviewed in the past in BEEBUG and other magazines. Pineapple claims that on the Archimedes the software runs four times faster than the BBC version. The package costs £63.25 inc VAT and p&p on 3.5" disc. Pineapple is at 39 Brownlea Gardens, Seven Kings, Ilford Essex IG3 9NL or telephone 01-599 1476.

ARMADILLO APPEARS

Armadillo Systems Ltd has announced that their Archimedes Sound Sampler is now available at a cost of £126.50 inc. VAT. Contact Armadillo at 17 Glaston Road, Uppingham, Leics. LE15 9PX or telephone (0572) 822499.

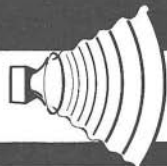
NEW SOFTWARE HOUSE

New software house, CCD Computer Services, has announced a range of products for the Archimedes. The Fortran Graphics Library adds over 90 graphics subroutines for use with the Acorn Fortran 77 compiler and costs £49.50. The XED Text-File Editor is designed specifically for creating and editing text files such as Fortran, Pascal, and C source code. Full screen editing is possible using the mouse in 40, 80 or 132 column mode, but with full access to star commands to compile and run programs, enter Basic etc. XED costs £19.50. Lastly, the Printer Module provides the Archimedes with a fast and flexible printer buffer for just £12.50. All prices are fully inclusive. CCD are at 71 Marlborough Park Avenue, Sidcup, Kent DA15 9DL or telephone 01-302 5427.

RESOURCES FOR THE ARCHIMEDES

A suite of programs of interest to all ex-BBC micro users with an Archimedes has been produced by Resource, a consortium of four local education authorities. The utilities are intended to assist in the conversion of BBC programs to run on an Archimedes. Examples of the utilities available include BBCscan for checking tokenised Basic, PicConv for converting screen dumps and GXR Sprites, and UdgConv for converting User Defined Characters. The package originates through Acorn but is being marketed by Resource with Acorn's agreement for £18.34 inc. VAT and p&p. Resource also expects to release an I/O podule shortly, with 16 bit input and output ports and two high speed (100KHz) A to D converters. Price including supporting software is expected to be around £120. For further information contact Nigel Hunter at Resource on (0302) 63800 or write to Exeter Road, Off Coventry Grove, Doncaster DN2 4PY.

RU



THE RISC USER VOICE GENERATOR

by Mark Davis

Tired of the Archimedes sound repertoire ? Then use this relocatable module to create new voices of your own, or simply make use of the seven new voices incorporated in the listing below.

The Archimedes sound system is a highly sophisticated one, but the nine resident voices do little justice to its power or flexibility. The accompanying program is intended to put some of that power more readily into the user's hands. It creates a relocatable module containing any number of user-defined voices up to the operating system maximum of 32. Each is defined prior to the creation of the module in terms of a formula, which may contain any number of sine waves, random elements etc, plus volume and pitch data. This enables the generation of virtually any sound. Once the module has been created, and loaded into the RMA, the user need only allocate the new voices to a sound channel number before obtaining full control over each voice via the usual SOUND command. You can even use the new voices in conjunction with the Welcome Music Editor.

USING THE PROGRAM

You should begin by carefully typing in the program in listing 1. This should be saved away before it is run. If it runs successfully you should see messages on the screen indicating the assembly of each of the 7 new voices. The newly created module is then automatically saved to disc (under the name UserVoices). To test it out, issue the following commands directly after assembling the module:

```
*RMLoad UserVoices
*ChannelVoice 2 2
VOICES 4
SOUND 2, -15, 100, 50
```

You should now hear a warbling sound which dies away after a second or so.

To test the whole set of voices, you could type in listing 2. This allocates and displays the 7 voices, then plays a sort of rhythm by cycling through the different voices at increasing frequency. If you press Escape (the crashing

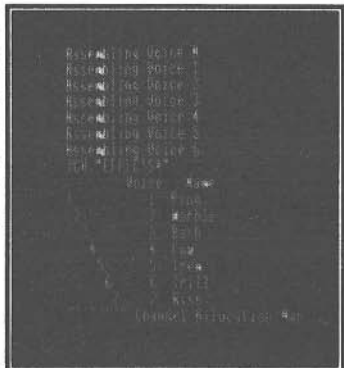
noise is caused by what appears to be a bug in the OS), you will see that function keys f1 to f7 play the seven new sounds at a SOUND pitch of 100.

CREATING YOUR OWN VOICES

The program has been written so that it is relatively easy to customise to meet individual requirements. The data for each sound is given in the early lines of the program between sets of dashed lines. After setting the variable TotalVoices% on line 90 to the number of voices to be created, you should give the formula for your first sound and its name as the two parameters of the procedure PROCfill.

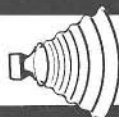
Line 200 is an easy example to follow. It sets up the voice named "Ping" with the formula "SIN(Z)". You will see that the formula used for "Warble" on line 240 contains an element of third harmonic (using SIN(Z*3)), and that scaling has been applied to give the harmonic one quarter of the volume of the fundamental.

The actual numbers used here are not important, only their relative values, since they are automatically scaled. Line 310 uses a random component in the voice formula, and you should note here that formulae containing randoms are not automatically scaled, and that in such cases the result of the formula must always be within the range -125 to +125. The variable Z ranges from 0 to 2π , and X from 0 to 255, both in a total of 256 steps for each voice.



THE ENVELOPES

The amplitude and frequency envelopes for each voice are created by making one or more calls to the procedures PROCaenv and PROCfenv. Both procedures have two parameters: the number of steps and the size of step; and the final call to each of the two



THE RISC USER VOICE GENERATOR

procedures for any given voice must contain -1 as the second parameter. The Ping voice provides a very simple example. PROCaenv is called twice, once with parameters 127 and 1, giving a single step up to maximum volume (127=maximum, 0=minimum). The second call (-4,-1) gives a decay down to zero in steps of -4 units. The unit of time, or step width, for both frequency and amplitude is one centi-second. Since we require the pitch of the Ping voice to be steady, PROCfenv is called just once with parameters 0,-1.

The data for Warble is more complex. PROCaenv is called three times with parameters (127,1), (0,32) and (-1,-1). The first takes the sound up to full volume in a single step. The second holds that volume for 32 steps, and the third releases it to zero at one step per centi-second. PROCfenv, as you can see, is called 41 times. The FOR loop repeatedly calls the procedure to give 2 steps which drop the pitch by 24, followed by 2 which raise it by 16. The final call completes the envelope with a further drop in pitch.

As you can appreciate, there is virtually no limit to the sounds which can be produced using this system. The only problem likely to arise is that with a very complex voice, you might run out of allocated memory. This is easily resolved by increasing the two constants on lines 140 and 150 from their current value of 500 to something greater.

Other sounds to experiment with

Square wave	(X>127)+.5
Sawtooth wave	X-128

```

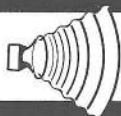
10 REM >Voice2Gen4
20 REM Program User Voice Generator
30 REM Version A 1.54
40 REM Author Mark J Davis
50 REM RISC User March 1988
60 REM Program Subject to Copyright
70 :
80 DIM start% &10000
90 TotalVoices%=7:REM User-defined
100 :
110 code2%=start%+300+20*TotalVoices%
120 wavetable%=code2%+TotalVoices%*600

```

```

130 aenvtable%=wavetable%+TotalVoices%
*256
140 fenvtable%=aenvtable%+500
150 end%=fenvtable%+500
160 voice%=0:E%=0:F%=0
170 :
180 MODE0:PROCRS:PROCmodass
190 REM-----
200 PROCfill("SIN(Z)", "Ping")
210 PROCaenv(127,1):PROCaenv(-4,-1)
220 PROCfenv(0,-1)
230 REM-----
240 PROCfill("80*SIN(Z)+20*SIN(Z*3)", "
Warble")
250 PROCaenv(127,1)
260 PROCaenv(0,32):PROCaenv(-1,-1)
270 FORX=1TO20
280 PROCfenv(-24,2):PROCfenv(16,2)
290 NEXT:PROCfenv(-8,-1)
300 REM-----
310 PROCfill("RND(250)-125", "Bash")
320 PROCaenv(127,1):PROCaenv(-4,-1)
330 PROCfenv(0,-1)
340 REM-----
350 PROCfill("80*SIN(Z)+20*SIN(Z*3)+20
*SIN(Z*7)", "Pow")
360 PROCaenv(127,1):PROCaenv(-2,-1)
370 PROCfenv(-64,-1)
380 REM-----
390 PROCfill("80*SIN(Z)+20*SIN(Z*3)+20
*SIN(Z*7)", "Trem")
400 PROCaenv(127,1)
410 FORX=1TO10:PROCaenv(-4,5):PROCaenv
(4,5):NEXT
420 PROCaenv(-4,-1)
430 PROCfenv(0,-1)
440 REM-----
450 PROCfill("(X>127)+.5", "Trill")
460 PROCaenv(127,1):PROCaenv(-1,-1)
470 FORX=1TO30:PROCfenv(-100,1):PROCfe
nv(100,1):NEXT
480 PROCfenv(-1,-1)
490 REM-----
500 PROCfill("SIN(Z)", "Rise")
510 PROCaenv(127,1):PROCaenv(-1,-1)
520 PROCfenv(100,-1)
530 REM-----
540 OSCLI"SAVE UserVoices "+STR$~start
%+" "+STR$~end%
550 *SETTYPE UserVoices &FFA
560 *STAMP UserVoices
570 END
580 :

```



THE RISC USER VOICE GENERATOR

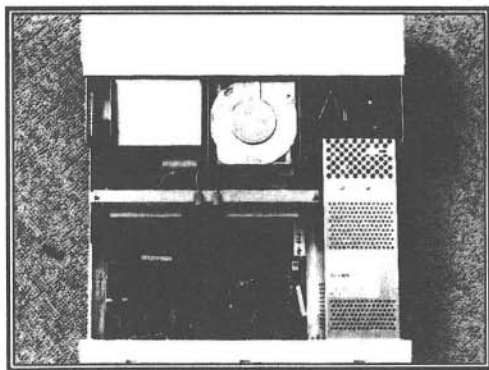
```
590 DEFPROCmodass
600 FORPASS=0TO2STEP2
610 P%=start%
620 [OPT PASS
630 .mstart
640 EQU0:EQU0 init-mstart
650 EQU0 fin-mstart:EQU0:EQU0 title-m
start
660 EQU0 help-mstart:EQU0
670 .title EQU0"UserVoiceGenerator"
680 EQU0:ALIGN
690 .help EQU0"Voices 1.00 (16 Jan 198
8)"
700 EQU0:ALIGN
710 .nos EQU0:EQU0:EQU0:EQU0
720 EQU0:EQU0:EQU0:EQU0
730 .init
740 STMFD R13!,{R14}
750 ADR R0,code2%:MOV R2,R0:MOV R1,#0
760 ADR R3,nos:SWI &40183:STRB R1,[R3,
#1]
770 ]
780 IF TotalVoices%>1 THEN
790 FORX=2TOTotalVoices%:[OPTPASS
800 ADD R2,R2,#600:MOV R0,R2:MOV R1,#0
810 SWI &40183:STRB R1,[R3,#1]
820 ]:NEXT
830 ENDIF
840 :
850 [OPTPASS
860 LDMFD R13!,{R15}
870 .fin
880 STMFD R13!,{R14}
890 MOV R2,#0:ADR R3,nos
900 .flp LDRB R1,[R3,R2]:SWI &40184
910 ADD R2,R2,#1:CMP R2,#32:BCC flp
920 LDMFD R13!,{R15}
930 ]
940 NEXT
950 ENDPROC
960 :
970 DEFPROCwavass(name$,w%)
980 FORPASS=0TO2STEP2
990 P%=code2%+w%*600
1000 [OPT PASS
1010 .voicecode
1020 B fill:B fill:B gateon:B gateoff
1030 B instance:LDMFD R13!,{PC}
1040 LDMFD R13!,{PC}
1050 EQU0 voicecode-voicecode
1060 .logampptr EQU0
1070 .wavebase EQU0 wavetable%+w%*256-w
avebase
1080 .aenvbase EQU0 aenvtable%+E%
1090 .fenvbase EQU0 fenvtable%+F%
1100 .waveinc EQU0 wavetable%+w%*256-wa
vebase
1110 .aenvinc EQU0 aenvtable%+E%-aenvba
se
1120 .fenvinc EQU0 fenvtable%+F%-fenvba
se
1130 .instance
1140 STMFD R13!,{R0-R4}
1150 MOV R0,#0:MOV R1,#0
1160 MOV R2,#0:MOV R3,#0
1170 MOV R4,#0:SWI "Sound Configure"
1180 LDR R0,[R3,#12]:STR R0,logampptr
1190 LDMFD R13!,{R0-R4,PC}
1200 .aenvaddr EQU0 0
1210 .fenvaddr EQU0 0
1220 .gateon
1230 ADR R0,wavebase:LDR R1,waveinc
1240 ADD R1,R1,R0:STR R1,[R0]
1250 ADR R0,aenvbase:LDR R1,aenvinc
1260 ADD R1,R1,R0:STR R1,[R0]
1270 ADR R0,fenvbase:LDR R1,fenvinc
1280 ADD R1,R1,R0:STR R1,[R0]
1290 LDR R0,wavebase:STR R0,[R9,#16]
1300 LDR R0,logampptr:STR R0,[R9,#20]
1310 LDR R0,aenvbase:STR R0,aenvaddr
1320 LDR R1,[R0]:STR R1,[R9,#28]
1330 LDR R0,fenvbase:STR R0,fenvaddr
1340 LDR R1,[R0]:STR R1,[R9,#24]
1350 LDR R0,[R9]:BIC R0,R0,#&7F
1360 STR R0,[R9,#0]
1370 .fill
1380 LDMIA R9,[R1-R8]
1390 AND R1,R1,#&7F
1400 AND R0,R8,#&FF:TST R8,#&100
1410 ADDEQ R1,R1,R0:CMP R1,#&7F
1420 MOVHI R1,#&7F:TST R8,#&100
1430 SUBNE R1,R1,R0:CMP R1,#&7F
1440 MOVHI R1,#0:SUBS R8,R8,#&10000
1450 LDRMI R0,aenvaddr
1460 LDRMI R8,[R0,#4]!
1470 STRMI R0,aenvaddr
1480 AND R0,R7,#&FF:TST R7,#&100
1490 ADDEQ R2,R2,R0:SUBNE R2,R2,R0
1500 SUBS R7,R7,#&10000
1510 LDRMI R0,fenvaddr
1520 LDRMI R7,[R0,#4]!:STRMI R0,fenvadd
r
1530 LDR R0,[R9]:BIC R0,R0,#&7F
1540 ORR R1,R0,R1:STMIA R9,[R1-R8]
1550 AND R1,R1,#&7F
```

continued on page 30

ARCHIMEDES 440 REVIEW

The Archimedes 440 is now in the shops in limited numbers at least. Lee Calcraft takes a look at what this top of the range machine has to offer.

The entry price of the Archimedes 440 is around £2600. This is almost three times the cost of an 310, so what extra features do you get for your money? Externally the difference between the 440 and the 300 series machines is minimal. All make use of identical system boxes, and the only difference in the keyboards is that the 440 has grey coloured function keys instead of the distinctive red colour used on the 330 series. At the rear of the machine there is again very little difference. All plugs, sockets and panels are identical except that the phono video socket on the 300 series has been replaced on the 440 by a pair of BNC sockets used for the attachment of a special high frequency mono monitor (video and sync).



Interior of A440 showing hard disc

Internally of course the machine looks quite different. It has a completely new PCB designed to accommodate 4 Mbytes of RAM. The strict upper limit for the 300 series is 1 Mbyte. There are three other obvious differences in the internal hardware. The machine houses a 20 Mbyte hard disc (of which more later), a small cooling fan, and a four-slot backplane. A backplane is essentially a board extension socket. The 300 series machines may be upgraded to take a two-slot backplane at a cost of around £40, but the 400 series machines provide four as standard. Of the four, slot number 2 may be used to hold a co-processor, and Acorn plan to have a floating-

point co-processor board on the market by the last quarter of 1988. The 300 series machines, by contrast, have no co-processor interface, and may not be upgraded to take one.

The manuals and firmware of the 440 are again identical to those supplied with 300 series machines. The 440 is currently supplied with Arthur 1.2, and with Basic V version 1.02. Its processor runs at the same speed as that of 300 series machines, and although the RAM chips in 400 series machines are somewhat faster, they are not clocked at a faster rate. The machines therefore perform identically, except where storage operations are involved.

HARD DISC

The 440's internally mounted 20 Mbyte Tandon hard disc provides a welcome increase in both storage speed and capacity. It is a delight to use, and runs more quietly than many hard disc machines. The accompanying speed tests give some idea of its performance on 3 different types of test: saving and loading 80K screens using the *ScreenSave and *ScreenLoad commands, saving and loading an 80K block of RAM using *SAVE and *LOAD, and finally the PCW "Store" benchmark, which involves writing a 20 byte string to a file 1000 times in succession. I have given comparison timings for both 600K and 800K floppies. And you may well be more surprised by the relatively slow speeds attained with the 600K format ADFS floppy than with the high speed of the Winchester. I should add that the 300 and 400 series machines give identical timings for all floppy disc tests.

Test	Floppy		Hard Disc
	600K	800K	
*ScreenSave (80K)	85.7	25.3	10.0
*ScreenLoad (80K)	40.8	18.0	5.6
*SAVE (80K)	5.5	3.5	0.4
*LOAD (80K)	5.2	3.2	0.3
Store Benchmark	18.9	6.4	3.1

Fig 1. A440 Timings (times in seconds)

NEW SCREEN MODES

Archimedes 300 series machines boast 21 screen modes, of which the top three (i.e. modes 18, 19 and 20) require a special multi-sync monitor. The 440 has two extra screen modes, labelled 22 and 23 (number 21 is "reserved for future expansion"). The two new modes are mono only, and require a further "special" monitor (in this case, one with a 96MHz line scan). Of the two, mode 23 is a text only mode, giving 144 characters on 54 lines (though you need a special 8x16 font for this). Mode 22 more usefully combines text (at 160 characters by 122 lines) with graphics, where it provides a resolution of 1280x976. This is very close to the 1280x1024 graphics units used on all versions of the BBC micro. The 48 missing vertical lines are taken equally from the top and bottom of the screen. As you will appreciate this is a very impressive resolution, and will be ideal for CAD and other graphics use, though lack of a special monitor prevented me from testing it out.

RAM CONFIGURING

When you enter Basic on a default-configured 440 you are presented with the message:

Starting with 3698940 bytes free

There is obviously room here for some quite long programs! Exactly how much RAM is allocated depends, as it does with the 300 series, on the *CONFIGURE options. On the 300 series these allocations are all made in page units of 8K (except for FontSpace, which uses 4K pages). In 400 series machines, the size of page used for the *CONFIGURE command is four times as large (except for FontSpace which remains at 4K). Thus if you execute:

```
*CONFIGURE ScreenSize 10
```

you will allocate 320K of RAM to screen use on a 440.

The default *CONFIGURE options on the 440 are adjusted accordingly, and are given in the accompanying table. Software designed to run on the full range of Archimedes machines will need to take account of the new configured page sizes. Unfortunately there is no very easy way of distinguishing between the various

members of the Archimedes range. The normally useful INKEY(-256) call gives the same result for all members, and is only useful for distinguishing between an Archimedes and earlier BBC micros. To use this test, type:

```
PRINT INKEY (-256)
```

All Archimedes machines give the result 160. The test given in listing 1 will however distinguish between the two Archimedes machine series. It reads the MEMC status register, and prints out the machine series number by checking whether the memory page size is 8K (300 series) or 32K (400 series).

```
10 REM Tests 300 or 400 series
20 PRINT300-100*FNfour;" Series machi
ne"
30 END
40 :
50 DEFFNfour
60 SYS &1A ,0,0 TO reg
70 =(reg AND 8)=8
```

Function	Config	Allocation
Font Size	6	24K
Screen Size	0	160K
RAM Filing	0	0
System Size	0	32K
RMA Size	2	64K*
Sprite Size	1	32K

*RMA Size 64K nominal. With all resident modules engaged, the actual space allocated to the RMA is 192K.

Fig 2. A440 default RAM allocations

CONCLUSION

The added RAM, hard disc, ultra high resolution modes and the potential offered by the co-processor interface make the 440 a highly desirable piece of kit. Its high price will of course mean that only educational and business establishments will in general be able to justify its purchase. Home users who feel the need for the 4 Mbytes of RAM and high resolution offered by the 400 series machines will need to wait for the emergence of the 410 in the spring. This has no hard disc, and has 1 Mbyte of RAM upgradable to 4 Mbytes, but costs just £1608.85 for the entry level system.

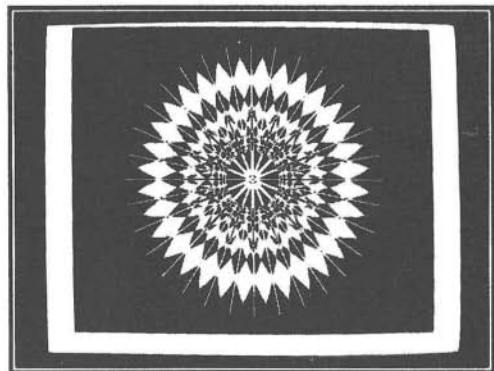
ARCHIMEDES VISUALS

We bring you another set of visually interesting routines accomplished with the minimum of code.

PATTERN GENERATOR

by L. Trail

This program generates an infinite number of coloured patterns. Each is based on a circle shape (see figure), and every second or so the circle is swept about the centre with two sets of ellipses. Each sweep is of random colour and plotting mode, thus building up an infinite variety of patterns. The program may be frozen by pressing the space bar. Pressing any other key will release it. Similarly, holding down any key apart from the space bar will speed it up.



Listing 1

```
10 REM >PattnGen2
20 REM Pattern Generator
30 REM by L. Trail
40 :
50 MODE 15:OFF
60 G=RND(-TIME)
70 ORIGIN 640,512
80 GCOL 185:CLG
90 VDU 19,0,24,240,240,128
100 :
110 REPEAT
120 Z%=RND(100)
130 M%=450
140 FOR E=11.25 TO 180 STEP 11.25
150 ELLIPSE FILL 0,0,Z%,M%,RAD(E)
160 NEXT
```

```
170 Z%=RND(100):M% = 250
180 FOR E=22.5 TO 180 STEP 22.5
190 ELLIPSE FILL 0,0,Z%,M%,RAD(E)
200 NEXT
210 GCOL RND(7),RND(64) TINT RND(4)
220 T=INKEY(100):*FX15
230 IF T=32 REPEAT UNTIL GET<>32
240 UNTIL FALSE
250 END
```

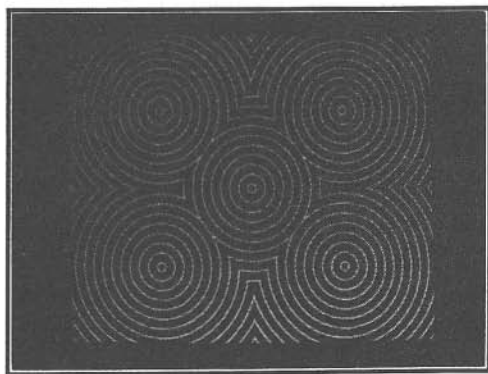
PALETTE-CHANGE EFFECTS

by Barry Christie

The programs in listings 2 and 3 make use of a continuously changing palette to create hypnotic and extremely smoothly moving effects. Listing 2 gives a five-centred whirlpool in blue, while listing 3 creates a moving arrow-head effect.

Listing 2

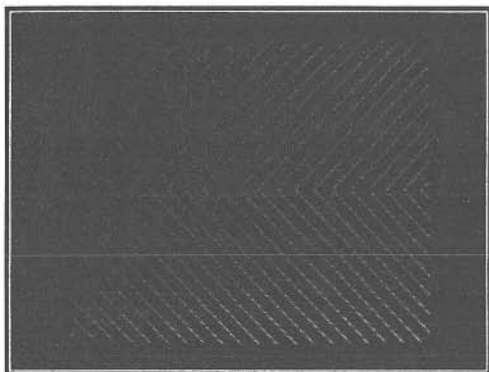
```
10 REM >Whirls1
20 REM Moving Whirlpools
30 REM by Barry Christie
40 :
50 MODE 12
60 FOR I%=205 TO 0 STEP -1
70 GCOL I% MOD 16
80 CIRCLE FILL 320,256,I%*2
90 CIRCLE FILL 960,256,I%*2
100 CIRCLE FILL 320,768,I%*2
```



```

110 CIRCLE FILL 960,768,I%*2
120 CIRCLE FILL 640,512,I%*2
130 NEXT
140 REPEAT
150 FOR I%=0 TO 15
160 WAIT
170 FOR J%=1 TO 7
180 COLOUR I%+J%,0,0,J%*16
190 COLOUR I%+16-J%,0,0,J%*16
200 NEXT:NEXT:UNTIL FALSE
210 END

```



Listing 3

```

10 REM >Arrows1
20 REM Hypnotic Steel Arrows
30 REM by Barry Christie
40 :
50 MODE 12
60 FOR I%=446 TO 0 STEP -1
70 GCOL I% MOD 16
80 PLOT 4,0,512+I%*4
90 PLOT 4,I%*4,512
100 PLOT 117,0,512-I%*4
110 NEXT
120 REPEAT
130 FOR I%=0 TO 15
140 WAIT
150 FOR J%=1 TO 7
160 COLOUR I%+ J%,J%*16,J%*16,J%*16
170 COLOUR I%+16-J%,J%*16,J%*16,J%*16
180 NEXT:NEXT:UNTIL FALSE
190 END

```

MOUSE DRIVEN COLOURED DISCS

by Jeroen Boomgaardt
and Frits Steenmeijer

The program in listing 4 is similar to one published in RISC User Issue 2. It uses the mouse to paint with randomly coloured discs. Use *select* to draw, *menu* to increase the disc size, and *adjust* to reduce it. The latter two buttons when pressed simultaneously, clear the screen, while pressing all three quits the program.

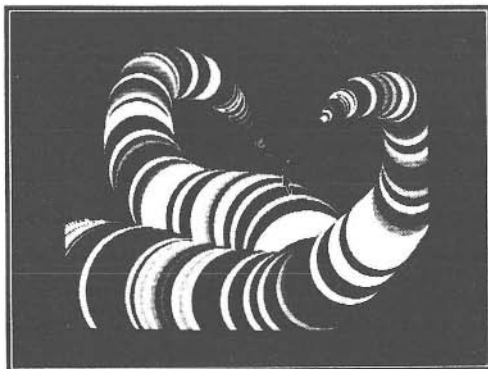
Listing 4

```

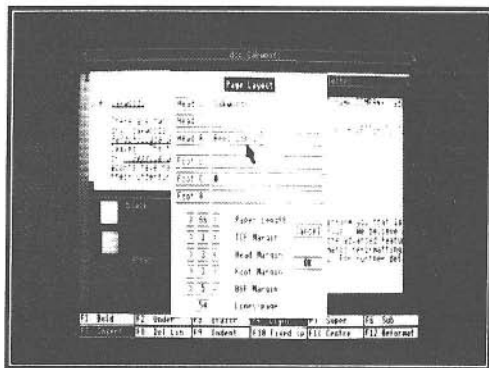
10 REM >Circles2
20 REM Mouse driven disc effects
30 REM by Jeroen Boomgaardt
40 REM and Frits Steenmeijer
50 :
60 MODE 15:OFF
70 R%=75:*POINTER
80 REPEAT:REPEAT
90 MOUSE X%,Y%,B%
100 UNTIL B%<>0
110 GCOL RND(63) TINT 64*RND(4)
120 IF B% AND 4 WAIT:CIRCLE FILL X%,Y%,R%
130 IF (B% AND 2)AND R%<1000 R%+=2
140 IF (B% AND 1)AND R%>1 R%-=2
150 IF B%=3 CLS
160 UNTIL B%=7:ON
170 END

```

RU



A page layout menu allows details of headers and footers to be specified, and the sizes of header and footer margins and page length. Automatic page numbering is provided for. The layout of the text on the page is determined by the use of rulers. When a new ruler is to be created or an existing one edited a ruler menu is displayed. Rulers may be up to 150 characters in length.



Tabs can also be adjusted on the ruler itself using the mouse pointer, and either a text tab or a decimal tab may be specified. It would have been nice to position tabs more precisely than whole character positions (1st Word Plus appears not to support proportionally spaced text at all), and to have text centred on a Tab as an option rather than just left justified.

In use, pressing Tab moves to the next tab position but results in *hard* spaces being inserted in the text which must then be deleted individually if the need arises. Much more useful is what 1st Word Plus calls an *Indent* which inserts a variable width space. This behaves more like a conventional Tab key and is the obvious thing to use for tables.

Another feature of page layout is the use of hyphenation. This is optional, but if switched on, 1st Word Plus will try to hyphenate long words if necessary. The hyphenation menu allows you to determine exactly where, if at all, hyphenation should take place. This is about the minimum that can be provided for hyphenation. One might have anticipated more control with a full hyphenation dictionary and a customised exceptions table.

SPELLING CHECKER

Before the spelling checker can be used the dictionary must be loaded into memory. This of course reduces the number of pages of text that are possible in memory. Acorn says that 150 pages of 250 words each (small pages!) are possible on an A310 without the spelling dictionary loaded. Spelling can be switched on for constant checking, or complete or part documents can be checked as required. Words not matched may be added to a supplementary dictionary, or browse mode selected to find a correct spelling.

GRAPHICS

A graphics mode may be selected and any mode 12 (or mode 20) picture may be loaded and positioned on the page. Because of the different aspect ratios between screen and printer this shows both the picture on-screen and the space it will ultimately occupy on the printed page. Pictures may be moved around a document and text may be superimposed upon the graphics. You can also do things like cut and paste a section of a document containing graphics. However, there appears to be no facilities for cropping or scaling graphics, so the facility is quite limited.

PRINTING

Print control appears to be excellent. Any set of consecutive pages may be printed, and multiple copies made as well if required. Best of all, you can select draft or NLQ mode from the same menu, an example of the well thought-out user-friendliness I referred to at the start.

MAIL-MERGE

This is effectively a separate package, and appears to be quite comprehensive though I was unable to try it out in the time available.

CONCLUSIONS

1st Word Plus certainly has far more features than any word processor on previous BBC micros, and the integration of a spelling checker, graphics and mail-merge in the one package is an agreeable step forward. It is also more flexible, with many more features than the free ArcWriter, though that will undoubtedly satisfy the needs of many. There are also many additional touches that I have not had the space to mention let alone describe - the use of footnotes for example. I do have some minor gripes as indicated, and others might emerge after more extensive usage.

At this stage I feel quite impressed by what has been achieved. At the expected price 1st Word Plus offers excellent word processing power. I hope, though, that Acorn will not let matters rest here, but will seek to provide further improvements and facilities for a future release and produce the best word processor in the world. Surely that's what the world's fastest micro really deserves.

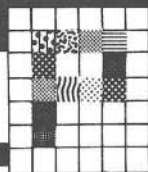
Product
Supplier

1st Word Plus
Acorn Computers Ltd,
Cambridge Technopark,
645 Newmarket Road,
Cambridge CB5 8PB.
Tel. (0223) 214411

Price

Not yet confirmed but expected
to be under £80 inc. VAT.

RU



A FULL-SCREEN PIXEL EDITOR

by Barry Christie

Use this pixel editor to edit full-screen images created in modes 13 and 15, or use it as a powerful drawing tool to create your own pictures.

Although the Sprite Editor supplied on the Welcome disc provides pixel editing facilities, the accompanying program offers a number of advantages. The Pixel Editor presented here gives the user a much larger display area in edit mode, and provides three levels of zoom to further improve flexibility. It can also display the whole screen at the touch of a button, and allows the user to move around the screen at great speed. It also provides a flood fill option, and allows the user to "pick up" colours from an existing picture, rather than insisting that they be selected each time from the cumbersome palette display.

USING THE PIXEL EDITOR

First of all you should type in the accompanying program, and save it away. When it is run it will begin by checking whether you have sufficient configured screen RAM. The program uses dual screens to give fast switching between edit and full display modes, and will therefore need 160K of RAM if used in mode 13, and 320K for mode 15. Use ***CON.SCR.20** followed by Ctrl-Break to allocate 160K, or ***CON.SCR.40** to allocate 320K. If you are using a 305 it is just possible to get 320K of screen RAM, but you will need to configure all other options to zero, except for sprite space, which should be kept at 1. Users of other machines will also need to have at least 1 unit of sprite space.

Once the program initialises you will be presented with the option screen. This offers four possibilities: star commands, screen save, screen load or edit mode. To get straight into edit mode, click on "EDIT", and reply to the prompt to select a mode. Again, if there is insufficient RAM, you will be informed. If all is well you will find yourself in the editing screen.

THE EDITING SCREEN

The editing screen is split vertically into two sectors. The left-most two inches or so of the screen constitute the control area, while the

rest is occupied by the editing area. All functions in the control area are activated by the middle mouse button (the *menu* button), while in the main editing area, all three buttons are used as follows:

select	Draw in current colour
menu	Pick up new colour
adjust	Flood fill in current colour

The currently selected colour is displayed above the colour palette in the control area. Using the menu button on the displayed palette in the control area will pick up a new colour. Pressing the menu button in the *editing* area will also pick up the colour currently under the pointer. The flood fill is a very useful option, though you should beware of "leaks" in any area to be filled. To make sure that it is not used by accident it requires a *double click* of the adjust button.



The Edit Screen

The control area itself is split into three unequal sections. At the top appears a true size image of the area being edited. If you click the menu button here, you will switch to a display of the whole screen with a window marker. Moving the mouse allows you to move the window to any part of the screen to redefine the editing area. A second click of the menu button will take you back to editing mode with the new area selected.



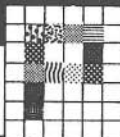
A FULL-SCREEN PIXEL EDITOR

Using the menu button on the central palette portion of the control area permits the selection of a new drawing colour. Finally, in the bottom left hand corner of the screen there is an eight-point scroller. Press the menu button on the appropriate point to scroll the drawing area in any one of eight directions; or in the centre to cycle through the three zoom options. Additionally there are two further options, but these are both initiated from the keyboard. Pressing the space bar will take you back to the option screen, while pressing Ctrl-Home will offer a screen clear. Finally, to quit the program, press Ctrl-Escape.

```
10 REM >PixelEdit9
20 REM Program Pixel Editor
30 REM Requires 160-320K of scn RAM
40 REM and 8K of sprite RAM
50 REM Version A 0.9
60 REM Author Barry Christie
70 REM RISC User March 1988
80 REM Program Subject to copyright
90 REM:
100 MODE 12:DIM B% &20
110 noerror=TRUE:validmode=FALSE
120 IF FNscrnsize<160 THEN PROCmessage
:END
130 mx=640:editx=0:my=512:edity=0
140 ON ERROR PROCerror
150 WHILE noerror
160 MODE 12:PROCOption
170 ENDWHILE
180 END
190 :
200 DEF PROCOption
210 PROCOptionscreen
220 MOUSE TO mx,my:*POINTER
230 REPEAT
240 REPEAT:MOUSE mx,my,mb:UNTIL mb<>0
250 chosen=mx DIV 320:areax=mx-chosen*
320:areay=(my+128) DIV 160
260 areachosen=areax>16 AND areax<304
AND areay=1:CLS
270 IF areachosen THEN
280 ON:*FX15
290 CASE chosen OF
300 WHEN 0:PROCstarcommand
310 WHEN 1:PROCloadpicture
320 WHEN 2:PROCsavepicture
330 WHEN 3:IF validmode THEN PROCeditp
icture ELSE PROCsetup:PROCeditpicture
```

```
340 ENDCASE:OFF
350 ENDIF
360 UNTIL areachosen
370 ENDPROC
380 :
390 DEF PROCstarcommand
400 INPUT "Please enter <STAR> command
... *"command$
410 PRINT:OSCLI(command$):PROCcontinue
420 ENDPROC
430 :
440 DEF PROCloadpicture
450 INPUT "Enter <filename> of picture
to LOAD ... "filename$
460 OSCLI("SCREENLOAD "+filename$):val
idmode=MODE
470 PROCvalidatemode
480 ENDPROC
490 :
500 DEF PROCvalidatemode
510 CASE validmode OF
520 WHEN 13:PROCassemble(320,4,&14000)
530 WHEN 15:PROCassemble(640,2,&28000)
540 OTHERWISE:validmode=FALSE
550 ENDCASE
560 ENDPROC
570 :
580 DEF PROCsavepicture
590 INPUT "Enter <filename> of picture
to SAVE ... "filename$:MODE validmode
600 PROCdriversdisplay(2,2):OSCLI("SCR
EENSAVE "+filename$)
610 PROCdriversdisplay(1,1)
620 ENDPROC
630 :
640 DEF PROCeditpicture
650 IF validmode THEN
660 PROCeditorscreen
670 REPEAT
680 MOUSE mx,my,mb
690 IF mx>260 THEN
700 CASE mb OF
710 WHEN 1:PROCfillingfunc
720 WHEN 2:PROCscrcolfunc
730 WHEN 4:PROCdrawingfunc
740 ENDCASE
750 ENDIF
760 IF mx<255 AND mb=2 THEN
770 CASE TRUE OF
780 WHEN my<255:PROCcontrolfunc
790 WHEN my>255 AND my<704:PROCmenucol
func
800 WHEN my>764:PROCeditingfunc
810 ENDCASE
820 ENDIF
```

A FULL-SCREEN PIXEL EDITOR

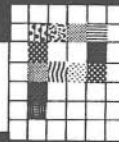


```

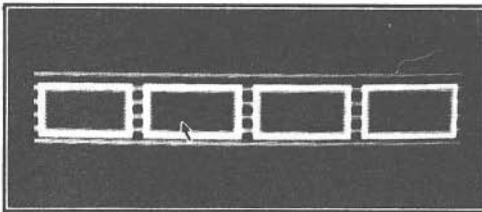
830 IF INKEY(-2) AND INKEY(-63) THEN P
ROCwipeoutfunc
840 UNTIL INKEY(-99)
850 ENDIF:ENDPROC
860 :
870 DEF PROCfillingfunc
880 REPEAT:MOUSE mx,my,mb:UNTIL mb=0
890 TIME=0:REPEAT UNTIL TIME>50
900 MOUSE mx,my,mb:IF mb<>1 THEN ENDPR
OC
910 GCOL POINT(mx,my)+128 TINT (TINT(m
x,my)):GCOL gcol TINT tint
920 PROCpictureplot(133):GCOL 128 TINT
0:PROCdisplayfunc(0,0)
930 ENDPROC
940 :
950 DEF PROCscrncolfunc
960 gcol=POINT(mx,my):tint=TINT(mx,my)
970 GCOL gcol TINT tint:RECTANGLE FILL
0,704,250,56
980 ENDPROC
990 :
1000 DEF PROCpictureplot(plotcode)
1010 PROCdriversdisplay(2,1)
1020 pixx=(mx-256) DIV dotx:pixy=?noofr
ows-(my DIV doty)-1
1030 PLOT plotcode, (editx+pixx)*dotsize
,1023-4*(edity+pixy)
1040 PROCdriversdisplay(1,1)
1050 ENDPROC
1060 :
1070 DEF PROCdrawingfunc
1080 MOUSE RECTANGLE 256,0,1023,1023
1090 REPEAT
1100 PROCpictureplot(69):PLOT 69,pixx*d
otsize,1023-4*pixy
1110 RECTANGLE FILL 256+pixx*dotx, (my D
IV doty)*doty, dotx-2, doty-4
1120 MOUSE mx,my,mb
1130 UNTIL mb<>4
1140 MOUSE RECTANGLE 0,0,1280,1024
1150 ENDPROC
1160 :
1170 DEF PROCcontrolfunc
1180 mx=mx DIV 84:my=my DIV 84:mb=0
1190 IF mx=1 AND my=1 THEN PROCzoomingf
unc ELSE PROCdisplayfunc(mx-1,-my+1)
1200 ENDPROC
1210 :
1220 DEF PROCmenucolfunc
1230 gcol=(mx DIV 64)*16+(my-256) DIV 2
8:tint=(mx DIV 16)*64
1240 GCOL gcol TINT tint:RECTANGLE FILL
0,704,250,56
1250 ENDPROC

1260 :
1270 DEF PROCeditingfunc
1280 PROCdriversdisplay(2,2):MOUSE RECT
ANGLE 0,0,1024,768
1290 dotstyle=0:delay=INKEY(20)*FX 21,
9
1300 GCOL 3,63:PROCsquare
1310 REPEAT
1320 PROCsquare:MOUSE mx,my,mb:dotstyle
=(dotstyle+1) MOD 8:PROCsquare
1330 UNTIL mb=2
1340 PROCsquare:GCOL gcol TINT tint
1350 editx=mx DIV dotsize:edity=256-(m
y+256) DIV 4)
1360 PROCdisplayfunc(0,0):MOUSE RECTANG
LE 0,0,1280,1024
1370 ENDPROC
1380 :
1390 DEF PROCwipeoutfunc
1400 PRINT TAB(0,0)"CLEAR SCREEN! ... (
Y/N)?"
1410 REPEAT:ans$=GET$:UNTIL INSTR("YyNn
",ans$)
1420 IF INSTR("Yy",ans$) THEN
1430 PROCdriversdisplay(2,1):GCOL gcol+
128 TINT tint:CLG
1440 PROCdriversdisplay(1,1)
1450 ENDIF
1460 PROCdisplayfunc(0,0)
1470 ENDPROC
1480 :
1490 DEF PROCsquare
1500 VDU 23,6,&FCFC>>dotstyle|
1510 PLOT 4,mx,my:WAIT
1520 PLOT 21,mx+252,my:PLOT 21,mx+252,m
y+252
1530 PLOT 21,mx,my+252:PLOT 21,mx,my
1540 ENDPROC
1550 :
1560 DEF PROCzoomingfunc
1570 GCOL 56:PROCshowzoom:zoom=(zoom+1)
MOD 3
1580 GCOL 63:PROCshowzoom:GCOL gcol TIN
T tint
1590 dotx=(4*dotsize)<<zoom:cols=(bytes
row*0.2)>>>zoom
1600 doty=16<<zoom:rows=64>>>zoom:?blok
size=1<<zoom
1610 ?noofcols=cols:maxix=bytesrow-cols
:IF editx>maxix THEN editx=maxix
1620 ?noofrows=rows:maxiy=256-rows:IF e
dity>maxiy THEN edity=maxiy
1630 PROCdisplayfunc(0,0):delay=INKEY(2
0)
1640 ENDPROC

```



A FULL-SCREEN PIXEL EDITOR



The Option Screen

```

1650 :
1660 DEF PROCshowzoom
1670 RECTANGLE 80+zoom*16,80+zoom*16,84
-zoom*32,84-zoom*32
1680 ENDPROC
1690 :
1700 DEF PROCdisplayfunc(alterx,altery)
1710 editx+=alterx:IF editx<0 OR editx>
maxix THEN editx=-alterx
1720 edity+=altery:IF edity<0 OR edity>
maxiy THEN edity=-altery
1730 PROCdriversdisplay(2,1)
1740 PLOT 4,editx*dotsize,1023-edity*4:
PLOT 0,255,-255:*SGET temp
1750 PROCdriversdisplay(1,1):*SCHOOSE t
emp
1760 PLOT &ED,0,768:CALL enlarger:*SDEL
ETE temp
1770 ENDPROC
1780 :
1790 DEF PROCdriversdisplay(drivers,dis
play)
1800 OSCLI("FX 112,"+STR$(drivers)):OSC
LI("FX 113,"+STR$(display))
1810 ENDPROC
1820 :
1830 DEF PROCcontinue
1840 PRINT"Press <SPACE> to continue .
..":REPEAT UNTIL GET=32:*FX 21,9
1850 ENDPROC
1860 :
1870 DEF PROCeditorscreen
1880 MODE validmode
1890 PROCdriversdisplay(1,2):PROCshowco
ntrols:PROCshowcolours
1900 PROCdriversdisplay(1,1):PROCzoomin
gfunc :*POINTER
1910 ENDPROC
1920 :
1930 DEF PROCshowcontrols
1940 GCOL 128 TINT 192:CLG:OFF:GCOL 52
1950 RECTANGLE FILL 0, 0,251,251:GCOL
0
1960 RECTANGLE FILL 16,16,219,219

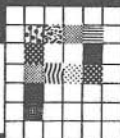
```

```

1970 PLOT 4,-68,124:PLOT 4,124,-68:PLOT
117,319,124:GCOL 56
1980 PLOT 4,0,124:PLOT 4,124,0:PLOT 117
,251,124:GCOL 0
1990 PLOT 4, 23,124:PLOT 4,124,23:PLOT
117,228,124
2000 RECTANGLE FILL 44,44,163,163:GCOL
56
2010 FOR zoom=0 TO 1:PROCshowzoom:NEXT
2020 GCOL 128 TINT 0
2030 ENDPROC
2040 :
2050 DEF PROCshowcolours
2060 mx=255:my=703:PROCmenucolfunc
2070 FOR G%=0 TO 63
2080 FOR T%=0 TO 3
2090 GCOL G% TINT T%*64
2100 RECTANGLE FILL (G% DIV 16)*64+T%*1
6,256+(G% MOD 16)*28,10,20
2110 NEXT:TINT:ENDPROC
2120 :
2130 DEF PROCOptionsscreen
2140 OFF:RESTORE:step%=1:barcol%=0
2150 VDU 19,0,24,0,0,240:COLOUR 0:COLOU
R 136
2160 FOR rgb%=0 TO 15:VDU 19,rgb%,16,0,
rgb%*16,240:NEXT
2170 FOR bars%=0 TO 55
2180 GCOL barcol%+7:RECTANGLE 0,bars%*4
,1280,0
2190 barcol%+=step%:IF barcol%<0 OR bar
col%>3 THEN step%=-step%
2200 NEXT
2210 FOR box%=0 TO 3
2220 FOR S%=0 TO 7
2230 GCOL 15-S%
2240 RECTANGLE FILL 16+box%*320+S%*4,32
+S%*4,286-S%*8,156-S%*8
2250 NEXT
2260 READ title$:PRINT TAB(box%*20+3,28
)title$
2270 NEXT
2280 VDU 28,0,24,79, 0:COLOUR 15:COLOUR
128:CLS
2290 ENDPROC
2300 :
2310 DATA "<STAR> command","<LOAD> pict
ure","<SAVE> picture","<EDIT> picture"
2320 :
2330 DEF PROCassemble(param1,param2,par
am3)
2340 bytesrow=param1:dotsize=param2:ban
ksize=param3
2350 DIM space 256:space!0=148:space!4=
-1:OFF

```

A FULL-SCREEN PIXEL EDITOR



```

2360 SYS &31,space,space+&10
2370 bankladr=space!&10
2380 bank2adr=bankladr+banksiz
2390 FOR assemble=0 TO 2 STEP 2
2400 P%=space
2410 [ OPT assemble
2420 ALIGN
2430 .enlarger
2440 LDR R0,smallpic
2450 LDR R1,largepic
2460 LDRB R2,noofrows;number of rows
2470 .rowsloop
2480 LDRB R3,noofcols;number of columns
2490 MOV R7,R0
2500 .colsloop
2510 LDRB R4,[R7],#&01;load and expand
2520 ORR R4,R4,R4,ASL #8
2530 ORR R4,R4,R4,ASL #16
2540 LDRB R5,bloksize
2550 .width
2560 MOV R8,R1
2570 LDRB R6,bloksize
2580 .depth
2590 STR R4,[R8],#bytesrow
2600 STR R4,[R8],#bytesrow
2610 STR R4,[R8],#bytesrow
2620 STR R4,[R8],#bytesrow
2630 SUBS R6,R6,#&01
2640 BNE depth
2650 ADD R1,R1,#4
2660 SUBS R5,R5,#&01
2670 BNE width
2680 SUBS R3,R3,#&01
2690 BNE colsloop
2700 ADD R0,R0,#bytesrow
2710 SUB R1,R8,#bytesrow*0.8-4
2720 SUBS R2,R2,#&01
2730 BNE rowsloop
2740 MOV R15,R14
2750 .transfer; this routine transfers
2760 LDR R0,smallpic; copy of the pic
2770 LDR R1,datadata; from
2780 MOV R2,R1 ;bank 1 to bank 2
2790 .transloop
2800 LDR R3,[R0],#4
2810 STR R3,[R1],#4
2820 CMP R0,R2
2830 BNE transloop
2840 MOV R15,R14
2850 .smallpic EQU bankladr
2860 .largepic EQU bankladr+0.2*bytesr
ow
2870 .datadata EQU bank2adr
2880 .noofrows EQU -1
2890 .noofcols EQU -1

2900 .bloksize EQU -1
2910 ]
2920 NEXT assemble
2930 CALL transfer
2940 ENDPROC
2950 :
2960 DEF PROCsetup
2970 REPEAT
2980 PRINT "Please select mode""Press
<1> for 13 or <2> for 15 ";
2990 REPEAT:key=GET-48:UNTIL key=1 OR k
ey=2
3000 IF NOT FNramtest(key) THEN PRINT''
"You must configure more RAM for this mo
de":PROCcontinue:CLS
3010 UNTIL FNramtest(key)
3020 validmode=2*key+11:CLS:OFF:PROCval
idatemode
3030 mx=640:editx=0:my=512:edity=0:noer
ror=TRUE
3040 IF key=1 THEN MODE 13 ELSE MODE 15
3050 PROCdriversdisplay(2,2):CLS
3060 ENDPROC
3070 :
3080 DEF PROCerror
3090 PROCdriversdisplay(1,1):CLS
3100 IF ERR=17 AND INKEY(-2) THEN
3110 noerror=FALSE
3120 ELSE
3130 REPORT
3140 IF (ERR AND &FF)<127 THEN PRINT" a
t line ";ERR ELSE PRINT
3150 IF ERR=17 PRINT"Use Ctrl-Esc to qu
it"
3160 VDU7:PROCcontinue
3170 ENDIF
3180 ENDPROC
3190 :
3200 DEFFNscrsnize
3210 !B%=150:!(B%+4)--1
3220 SYS&31,B%,B%+&10:=!(B%+&10)/1024
3230 :
3240 DEFFNramtest(size)
3250 =(160*size<=FNscrsnize)
3260 :
3270 DEFPROCmessage
3280 PRINT"Your machine must have at le
ast 160K of screen RAM for this program"
3290 PRINT"You will need 320K to use mo
de 15"
3300 PRINT"Please reconfigure with:"
3310 PRINT" *CONFIGURE SCREENSIZE 20"
"or *CONFIGURE SCREENSIZE 40"
3320 VDU7:ENDPROC

```



LOGICAL RAM TRACER

by Lee Calcraft

Use this program to ferret out every byte of logical RAM in your Archimedes. Whether you have a 305 or a 440 this program will display its memory map within a few seconds.

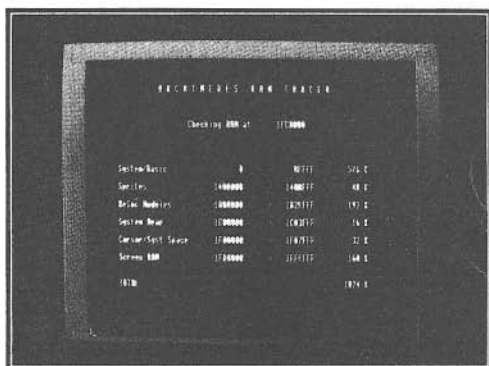
This logical RAM tracer was promised as a tailpiece to last month's article entitled "A memory map for Archie", and readers are referred to that article for further details on the logical mapping of RAM on the Archimedes. The present program uses a special operating system call "OS_ValidateAddress" to check for the presence of blocks of logical RAM. In other words it checks to see whether a given block of logical RAM is mapped to physical RAM or not. The program checks through the whole 32 Mbyte logical RAM area in its tireless quest for usable RAM, and displays the results.

Unfortunately, because of a bug in the operating system (even on OS 1.2), the call gives incorrect results when it gets near to the screen area, and a second OS call is used as a patch. When you run the program, you should see all current RAM allocations displayed, with a total allocation consistent with the size of your machine (i.e. 512, 1024 or 4096K). Please note that the displayed RAM for Basic also includes the RAM used for system space at &0000.

```

10 REM >RAMTracer1
20 REM Program Logical RAM Tracer
30 REM Version A 0.1
40 REM Author Lee Calcraft
50 REM RISC User March 1988
60 REM Program Subject to copyright
70 :
80 MODE12:OFF:contig=FALSE:Y=12:T=0
90 DIM B% &20:VDU19,0,24,128,128,240
100 PRINTTAB(18,3)"A R C H I M E D E S
    R A M   T R A C E R"
110 PRINTTAB(25,7)"Checking RAM at"
120 PROCcheckit
130 addr%=FNscrn:PROCstart
140 addr%=&2000000:PROCend
150 PRINTTAB(8,Y+1)"TOTAL"SPC(43)T;" K
"
160 PRINTTAB(0,31);:ON:END
170 :
180 DEFPROCcheckit
190 FOR addr%=0 TO &1FC8000 STEP &2000

```

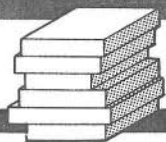


Typical RAM allocation on an A310

```

200 PRINTTAB(43,7)~addr%;SPC7
210 ram=FNcheck(addr%)
220 IF (NOT ram) AND contig PROCend
230 IF ram AND NOT contig PROCstart
240 NEXT:ENDPROC
250 :
260 DEFNcheck(addr%)
270 SYS &3A,addr%,addr%+1 TO ;flag
280 =(flag AND 2)=0
290 :
300 DEFPROCend
310 PRINTTAB(44,Y)~"-~(addr%-1)
320 ram%=(addr%-start%)/&400:T+=ram%
330 PRINTTAB(56,Y)ram%;" K"
340 Y=Y+2:contig=FALSE:ENDPROC
350 :
360 DEFPROCstart
370 PRINTTAB(8,Y)FNtxt,TAB(28,Y)~addr%
380 start%=addr%:contig=TRUE:ENDPROC
390 :
400 DEFNscrn
410 :!B%=148:!(B%+4)=--1
420 SYS &31,B%,B%+&10:=(B%+&10)
430 :
440 DEFNtxt:RESTORE
450 REPEAT READ start,text$
460 UNTIL addr%/&10000<=start:=text$
470 DATA 0,"System/Basic",&100,"RAM Fi
    ling",&140,"Sprites",&180,"Reloc Modules
    ",&1C0,"System Heap",&1F0,"Cursor/Syst S
    pace",&200,"Screen RAM"

```



WRITING A RELOCATBLE MODULE

By Lee Calcraff

Writing your own relocatable module made simple.

The relocatable module provides the programmer with a powerful means of implementing all manner of machine code applications and utilities in such a way that they appear to the user as if they were an integral part of the Archimedes' operating system. Indeed a great deal of the machine's firmware is constructed around this principle: except of course that, being in ROM, the resident modules are fixed rather than relocatable.

The aim of this article is to look at the way in which a simple relocatable module may be written. By way of illustration, the accompanying program creates a simple module which responds to the two commands ***Test** and ***Dummy**, and to various ***Help** calls. If you type in the program, and run it, you will see the code being assembled, and then a list of resident modules will be displayed. This list should include the test module as the last item. If you type ***HELP Modules**, you should again see the new module listed. This time there will be an accompanying version number, and the date of assembly. To confirm that the module functions correctly, you could also try typing ***Test**. This should produce the following display accompanied by a beep:

The response to ***Test**

HOW IT WORKS

The remainder of this article is devoted to a dissection of the program. Line 100 dimensions 4K of RAM in which the code will be assembled. For a real module you may need considerably more than this. Remember that each ARM instruction takes four bytes. Lines 110 to 130 set up the resident assembler to assemble your code at the address of the block dimensioned in line 100. Note that offset assembly has been employed (see Programmer's Reference Manual page 597) in such a way as to ensure that the code will be assembled as if it were to run at address &0000 (achieved by setting P% to 0 in line 120). Using the offset in this way makes all the address referencing within the module automatically refer to the start address of the code, as it must for relocatable code. Once assembly is

complete, line 1090 saves the assembled code, line 1100 sets the saved file to type &FFA (relocatable module), and line 1110 loads it into the RMA, from where it can be tested.

The most important part of the code presented here is the module header, and as you can see, this is extremely short. I have split it into two parts. The first 4 lines (lines 170 to 200) give the offset to four optional pieces of code, though since our simple module uses none of them, each offset is given as a zero value 32-bit word. For further details of the use of these options, the reader is referred to the Reference Manual page 366.

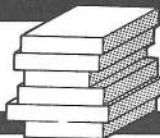
The last three lines of the header (lines 220 to 240) are concerned with the title of the module, the response to ***Help Modules**, and most important of all, the command table itself. All three are made use of in our example, although again, any of them could be replaced by a zero value 32-bit word if the user wished the operating system to ignore them.

MODULE TITLE STRING

Line 220 must contain either a zero 32-bit word or a 32-bit address pointing to a title string within the module. The address, like all those within the module must be given as an offset from the start of the module. But this is automatically handled by starting the module at address zero. We have inserted the variable "title" at this point. This refers to the label on line 290. The protocol for the title is just a string (containing no spaces) followed by a zero byte, and then the pseudo operator ALIGN to align the next instruction to a word boundary.

MODULE HELP STRING

The Help offset on line 230 works in a similar way. It refers to the "help" label on line 350. The protocol is identical (except that with this and all other strings in the module, spaces are permitted). The code consists of a string followed by a zero byte and an ALIGN directive. In this case I have used the machine's TIMES function to automatically incorporate the date on which the module was assembled.



WRITING A RELOCATABLE MODULE

THE COMMAND TABLE

Line 240 points to the module's command table. In our example this is located from line 400. It contains a set of five entries for each command name recognised by the module, the whole table being terminated with a zero 32-bit word (on line 600). The implementation of a command table of this kind will be most welcome to anyone who has written sideways ROM software for the model B or BBC Master, since it completely handles the parsing of the command. The programmer has simply to place his command name in the command table followed by a zero byte and an ALIGN directive. As you can see, the name "Test" appears at the head of the command table (on line 440). The second command "Dummy" appears on line 530, also followed by a zero byte.

After the zero-byte and ALIGN directive following each command name, there are four 32-bit words. The first points to the code to be executed when the command is recognised, the second contains a set of flags (of which more in a moment), the third optionally points to a message that the operating system will issue if the syntax of the command is given wrongly by the user, while the fourth points to a string of text which will be used as a response to the command *Help word, where "word" is one of the command keywords ("Test" or "Dummy" in this case).

THE COMMANDS THEMSELVES

Lines 470 and 560 point to the two commands themselves. The code for these begins at lines 830 and 980 respectively. They are similar, and we will restrict ourselves to a discussion of the code for "Test. There are two essential elements to this. The first line of code preserves the program counter on the stack with the instruction:

```
STMFd R13!, {R14}
```

The reason why the program counter is in register 14, and not 15 is that when the operating system passes the star command to the module, it performs a branch with link, preserving the program counter in register 14. To exit the module, and return to the language from which the star command was issued, we simply need to reinstate register 15 with the contents of the stack. This is achieved on line

930 with the instruction:

```
LDMFD R13!, {PC}
```

In the code which makes up the body of the command you should take care not to use R13 for any purpose other than as a stack pointer, otherwise you will lose your return address. R14 can be used as normal in Branch with link operations to perform subroutines within the module. But code in the module should not make reference to any absolute address, otherwise the routine will not be relocatable, and will crash. The code used to implement the command "Test is trivial, using SWIs 1 and 3 to write the string of text "The response to "Test", followed by character 7 to generate a beep.

THE RESPONSE FLAGS

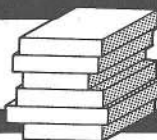
The second 32-bit word in the command table comprises a set of flags which tell the operating system about the star command which you have implemented. Their function is indicated in the accompanying table. Bytes 0 and 2 respectively give the minimum and maximum number of parameters which may be used with the star command. If the user supplies a number of parameters outside the acceptable range, he is prompted with the corresponding syntax message (set up on lines 490 and 580). In our example we have set the flags so that "Test must be given without parameters (max and min number =0), and "Dummy must have either one or two. The operating system, I should add, expects parameters to be separated by spaces, not commas.

It is hoped that, given this information about our skeleton module, users will be able to create customised modules of their own. For further details on writing modules (including the function of flag bytes 1 and 3, and the use of initialisation and finalisation code) the reader is referred to the Reference Manual part 2.

Byte	Contents
0	Min no of parameters (0-255)
1	OS_GTrans map (1st 8 params)
2	Max no of parameters (0-255)
3	3 bit flags

Command table flags

WRITING A RELOCATABLE MODULE



```

10 REM >TestMod8
20 REM Program Test Module
30 REM Version A 0.8
40 REM Author Lee Calcraft
50 REM RISC User March 1988
60 REM Program Subject to copyright
70 :
80 REM Generates a relocatable
90 REM module called RMtest
100 DIM R% &1000
110 FOR I=4 TO 7 STEP 3
120 P%=0:O%=R% :REM start code at &O
130 [OPT I
140 ;-----
150 ; Module Header
160 ;-----
170 EQU 0 ;startcode
180 EQU 0 ;initialise
190 EQU 0 ;finalise
200 EQU 0 ;service
210 ;
220 EQU title ;Title string offset
230 EQU help ;Help string offset
240 EQU command ;Command table offset
250 ;-----
260 ; Title & Help Strings
270 ;-----
280 ;Title of the module for *MODULES
290 .title
300 EQU "TestModule"
310 EQU 0
320 ALIGN
330 ;
340 ;Help message for *HELP MODULES
350 .help
360 EQU "Test Module"+CHR$9+"1.00 ("+"
MID$(TIME$,5,11)+")"
370 EQU 0
380 ALIGN
390 ;-----
400 ; Table of commands
410 ;-----
420 .command
430 ;Response to *TEST
440 EQU "Test" ;The command name
450 EQU 0 ;Zero terminator
460 ALIGN
470 EQU testcommand ;Address of code
480 EQU &00000100 ;Response flags
490 EQU testsyntax ;Invalid syntax
500 EQU testhelp ;Help text
510 ;
520 ;Response to *DUMMY
530 EQU "Dummy" ;The command name
540 EQU 0 ;Zero terminator
550 ALIGN
560 EQU dummycommand ;Address of code
570 EQU &00020001 ;Response flags
580 EQU dummiesyntax ;Invalid syntax
590 EQU dummyhelp ;Help text
600 EQU 0 ;table terminator
610 ;-----
620 ; Help and Syntax messages
630 ;-----
640 .testsyntax
650 EQU "Syntax: *Test (no params)"
660 EQU 0
670 ;
680 .testhelp
690 EQU "*TEST does very little"
700 EQU 0
710 ;
720 .dummiesyntax
730 EQU "Syntax: *Dummy <n> [<m>]"
740 EQU 0
750 ;
760 .dummyhelp
770 EQU "*DUMMY <n> does very little"
780 EQU 0
790 ALIGN
800 ;-----
810 ; Code for *TEST
820 ;-----
830 .testcommand
840 STMPD R13!,{R14} ;Preserve program
850 ;counter on stack
860 SWI 3
870 SWI 1
880 EQU "The response to *Test"
890 EQU 7
900 EQU 0
910 ALIGN
920 SWI 3
930 LDMFD R13!,{PC} ;Restore program
940 ;counter
950 ;-----
960 ; Code for *DUMMY
970 ;-----
980 .dummycommand
990 STMPD R13!,{R14}
1000 SWI 3
1010 SWI 1
1020 EQU "The response to *DUMMY"
1030 EQU 0
1040 ALIGN
1050 SWI 3
1060 LDMFD R13!,{PC}
1070 ;-----
1080 ]:NEXT
1090 OSCLI"SAVE RMtest "+STR$~R%+" "+STR$~P%
R$~P%
1100 OSCLI"SETTYPE RMtest FFA"
1110 *RMLOAD RMtest
1120 *MODULES

```

GETTING THE ARCHIMEDES ON LINE

**A brief survey of the state of comms on the Archimedes by Ian Burley,
News and Features editor of Micronet.**

The old BBC Model B was a superb comms tool. It had an easily accessible RS423 serial port - even from Basic, an 80 column screen for ASCII text, and perhaps most important of all, the full Teletext character set as standard.

The Archimedes carries on these traditions, and a number of companies are working on sophisticated comms packages for the Archimedes, though as this is written, none have been properly released, save one. The reason for the delay centres around the now infamous RS232 bug. Actually Acorn claims there are lots of bugs, and they're all the fault of the 6551 serial line chip they chose for the Archimedes. Unfortunately the latest release of Arthur (version 1.20) compounds RS232 problems even further, corrupting bytes being transmitted when bytes are being received. This effectively reduces the serial port to being half duplex! However, Acorn are on the verge of producing a relocatable module patch which they say will entirely cure the serial port problems.

The one Archimedes comms program actually available also happens to be free! It's ARCTerm, and was written by Hugo Fiennes, D.R.Boyd, and Brian Smith for the Public Domain. The program is written entirely in Basic, and offers quite adequate scrolling text handling, and a primitive viewdata mode. Hugo hates desktops and so he has deliberately ignored the mouse. ARCTerm boots up into scrolling text mode, and features a status area at the base of the screen, and help menus can be summoned via ALT-H.

Hayes command accessibility is provided, as is the ubiquitous XMODEM file transfer. The viewdata mode is disappointing, though improvements are promised. For example, once into viewdata you're cut off from the rest of ARCTerm, and you have to reset the machine to exit. There is also no telesoftware downloading facility. But the package is free! If you want to download ARCTerm, you can do so by dialling into Hugo's BB on (0458) 47608

(V21 through V22bis) using another machine with XMODEM capability.

Hugo is working on a commercial version of ARCTerm, to feature Kermit, ZMODEM, XMODEM and XMODEM CRC, XMODEM-1K, Megalink, full VT100, VT52, IBM ANSI colour, Teletype, CET viewdata downloading, and a built in host with its own programming language. But Hugo insists that even this won't be mouse driven!

Oddly enough, you can resort to using the 6502 emulator to get online to Prestel compatible systems using Watford's old Modem 84 ROM!

Also in the pipeline is "Deputy" written by Micky Spalter from Synchrotech. This is waiting until the RS232 problem is fixed. To be distributed by Dataphone, who make the new Demon Modems, Deputy does use the Archimedes WIMPs so the viewdata section won't use mode 7, but instead a screen mode to emulate viewdata characters whilst retaining desktop accessibility.

Deputy promises some fine features, e.g. a 300 telephone number store, Demon Modem auto baud-rate setting, an elapsed time clock, Hayes support, XMODEM and CET telesoftware, offline mailbox editing, a "chat-cache" for spooling text to, etc. This will eventually sell for £49.95.

Pete Gaunt, author of the excellent Modem Master for the Beeb, is working on a mouse driven package for the Archimedes, and this too will re-emulate mode 7 in 80 columns using the WIMPs. It's likely that this will eventually be released under the BBC Soft label, as was Modem Master.

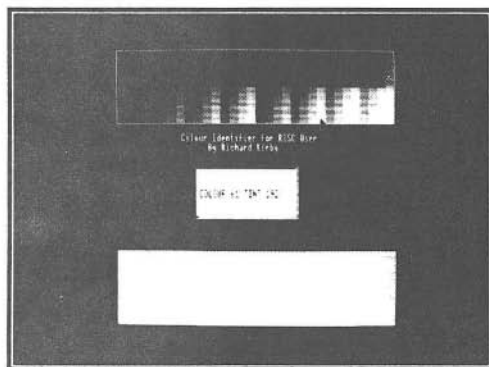
Finally, Acorn has a Dutch package called Acorn in the wings, but perhaps the one to look out for is BEEBUG's own package, as yet unnamed, to go with the keenly awaited BEEBUG internal modem for the Archimedes.

RU

AT THE END OF THE RAINBOW

This month we conclude our examination of the 256 colour modes with utilities by Richard Kirby and Keith McAlpine. Mike Williams sets the scene and explains the use of the palette registers to re-define colours.

This month I want to describe three useful programs all related to the 256 colour modes (modes 11, 13 and 15). We looked at this subject in some detail last month using the default palette and resulting colours. The first program, Colours2 by Richard Kirby, displays all 256 default colours on the screen in a way that makes choice of colour and corresponding GCOL and TINT parameters much simpler than either of last month's programs. Type the program in and save it to disc before you try it out. Use the mouse to select any of the 256 colours on the screen, and that colour will be displayed as a larger colour panel (allowing a better assessment of your choice) together with the GCOL and TINT numbers for reference.



```

10 REM          >Colours2
20 REM Program   Colour Identifier
30 REM Version   A1.3
40 REM Author    Richard Kirby
50 REM RISC User March 1988
60 REM Program   Subject to Copyright
70 :
80 MODE 15:OFF:*POINTER 1
90 ON ERROR GOTO 230
100 PROCcolours:PROCsquare
110 REPEAT:REPEAT
120 MOUSE x,y,button
130 UNTILbutton>0 AND ((x>128 AND x<11
52) AND (y>736 AND y<992))
140 colour=FNcolour(x,y)
150 tint=FNtint(x,y)

```

```

160 COLOUR 191 TINT 192
170 COLOUR 0 TINT 0
180 PRINTTAB(27,16)"COLOUR ";colour;"
TINT ";tint;SPC3
190 GCOL colour TINT tint
200 RECTANGLEFILL128,36,1024,256
210 UNTIL FALSE
220 :
230 ON ERROR OFF:MODE 3
240 REPORT:PRINT" at line ";ERL:END
250 :
260 DEFPROCcolours
270 col%=128
280 FOR rows%=1 TO 8
290 PRINTTAB(8,rows%);
300 FOR columns%=1 TO 8
310 FOR tint%=0 TO 3
320 COLOUR col% TINT tint%*64
330 PRINT SPC2;
340 NEXT:col%=col%+1:NEXT
350 IF rows%=4 THEN col%=128
360 col%+=8
370 NEXT:GCOL 63 TINT 192
380 RECTANGLE128,736,1024,256
390 RECTANGLE126,32,1028,264
400 ENDPROC
410 :
420 DEFNtint(x,y)=TINT(x,y)
430 :
440 DEFNcolour(x,y)=POINT(x,y)
450 :
460 DEFPROCsquare
470 GCOL 42 TINT 0
480 RECTANGLE FILL 400,400,400,200
490 GCOL63 TINT 0:MOVE408,408
500 MOVE792,408:PLOT85,792,592
510 GCOL0 TINT 0:MOVE408,592
520 MOVE792,592:PLOT85,408,408
530 GCOL63 TINT 192
540 RECTANGLEFILL420,420,360,160
550 COLOUR128 TINT 0:COLOUR63
560 PRINTTAB(23,10)"Colour Identifier
for RISC User"
570 PRINTTAB(29,11)"By Richard Kirby"
580 ENDPROC

```

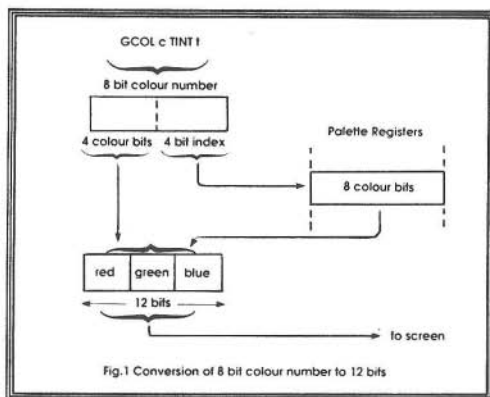
REDEFINING THE PALETTE

The palette registers (all 16) have been set to give a reasonable range of colours by

AT THE END OF THE RAINBOW

palette registers it is possible to select any of the Archimedes' 4096 colours as one of the 256 in current use. However, BE WARNED: understanding how this works is enough to make a strong man tremble, and it also destroys the colour relationship of the GCOL and TINT numbers described last month.

The reason for all the angst is that there are only 16 palette registers to control palette definitions for 256 colours. As a result, each palette register controls the appearance of 16 different shades. Changing a palette register will change ALL 16 related shades. In principle, the eight-bit colour byte stored in screen memory to specify the colour of a pixel is treated as two four bit nibbles. The top four bits (1 bit red, 2 bits green, 1 bit blue) are sent directly to the screen D to A converters (digital to analogue), while the bottom four bits form an index or pointer to a palette register which then supplies the remaining 8 bits of colour information needed (3 bits red, 2 green, 3 blue) as shown in the Figure below.



As a first demonstration of some of the detail involved, the following short program (TempPal3) uses palette register 1 to cycle through all 4096 colours. Three nested FOR loops (line 150) increment in turn the levels of blue, green and red using up to 16 parts each of these three basic colours ($16 \times 16 \times 16 = 4096$).

Lines 160 to 180 determine the top four bits of the 8 bit colour number (1 bit blue, 2 bits green, 1 bit red), and line 190 shows how the GCOL number (col%) is then determined from

these three values. Line 200 uses a VDU19 call to redefine palette register 1 with the appropriate amounts of red, green and blue (you can also use COLOUR 1,r,g,b), while in line 210 GCOL selects the colour and TINT selects the *palette register* to be used.

Type in and save the program. When you run it, it will cycle through all 4096 colours as indicated, and the palette definition and GCOL/TINT selection is displayed on screen. Use the cursor left and cursor right keys to slow down or speed up. Holding down Shift will allow you to single-step through the colours at any point in the sequence using the space bar.

Note that the function of the TINT parameter is totally different now (the concept of a white tint is only possible with the default palette). The program is not completely general, and as written will only work correctly for the redefinition of palette registers 0 to 3.

```

10 REM >TempPal3
20 REM Program 4096 Colour Display
30 REM Version A1.3
40 REM Author Mike Williams
50 REM RISC User March 1988
60 REM Program Subject to Copyright
70 :
80 MODE 13:OFF:ON ERROR GOTO 330
90 PRINTTAB(4,6)"4096 COLOURS in a 25
6 COLOUR MODE"
100 wait=10:GCOL 63 TINT 3<<6
110 RECTANGLEFILL476,188,328,264
120 PRINTTAB(20,12)"TINT 1<<6"
130 PRINTTAB(9,28)"<- slower faster
->"
140 PRINTTAB(13,30)"Shift to Pause"
150 FOR red%=0 TO 15:FOR grn%=0 TO 15:FO
R blu%=0 TO 15
160 blu3%=(blu% AND %1000)>>2
170 grn23%=(grn% AND %1100)>>2
180 red3%=(red% AND %1000)>>2
190 col%=(blu3%<<4)+(grn23%<<2)+(red3%
)
200 VDU19,1,16,red%<<4,grn%<<4,blu%<<4
210 GCOL col% TINT 1<<6
220 RECTANGLEFILL480,192,320,256
230 PRINTTAB(10,10)"VDU19,1,16,";red%<
<4;"";grn%<<4;"";blu%<<4;SPC6
240 PRINTTAB(10,12)"GCOL 0,";col%;SPC1
250 PRINTTAB(10,14)"Colour ";256*red%+
16*grn%+blu%+1

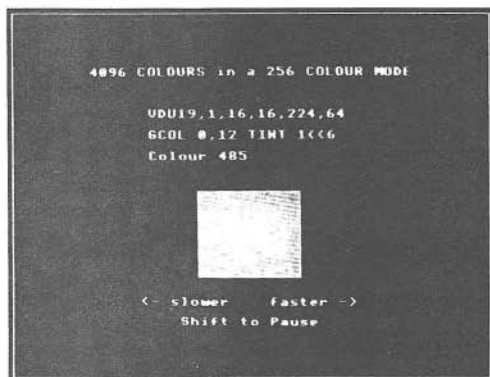
```

AT THE END OF THE RAINBOW

```

260 IF INKEY-26 THEN wait+=1
270 IF INKEY-122 THEN wait-=1
280 TIME=0:REPEAT UNTIL TIME>wait
290 IF INKEY-1 THEN G=GET
300 NEXT:NEXT:NEXT
310 END
320 :
330 ON ERROR OFF:MODE 12
340 REPORT:PRINT" at line ";ERL:END

```



The final program this month, and an excellent piece of programming by Keith McAlpine, demonstrates the redefinition of all 16 palette registers, and allows you to mix any colours you wish, displaying the appropriate palette re-definition and GCOL/TINT values. Again, you should save the program to disc after typing it in, and before running it.

The 256 default colours are displayed on the screen with each row of 16 colours corresponding to a palette register running from 0 to 15 down the screen. Use the mouse to select any colour and the corresponding palette definition and GCOL/TINT numbers will be displayed for reference. The screen also shows the amounts of red (range 0 to 7), green (0 to 3) and blue (0 to 7) which are the variable colour components stored in each palette register. Moving the mouse pointer across the colours as you hold down the select button will produce a constantly changing and updated display.

Use the mouse pointer to change the mix of red, green and blue in any palette definition, and watch how the corresponding row of

colours changes. You can always reset any palette register to its default value by clicking on the appropriate box. At any time you can choose to display (or print out) the current definitions of all 16 palette registers, and this information is automatically left displayed on the screen when you quit from the program.

Using the information provided by the program you should be able to redefine any colour and determine the corresponding COLOUR and GCOL/TINT statements. Remember though, the limitations. The 256 displayed colours are not all independent from each other. The best you can hope for is to define 16 really good colours, one for each palette; anything else is a bonus.

ENDPOINT

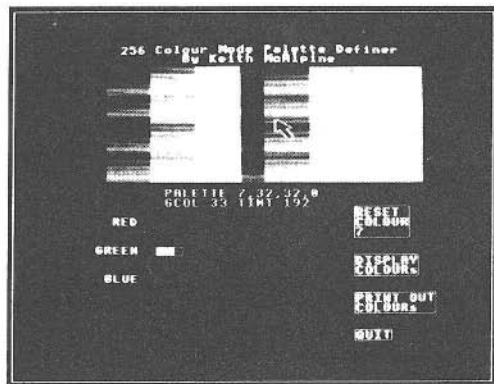
I have deliberately kept the explanations as short and as simple as possible. Attempting to explain fully how the palette registers and the rest work in the 256 colour modes would be a very substantial task requiring several pages of RISC User. What I set out to do instead, with the help of the programs from Richard Kirby and Keith McAlpine, is to provide you with the means to experiment with colour redefinition in a reasonably constructive way.

```

10 REM >PalDef256
20 REM Program 256 Colour palette d
efiner
30 REM Version A1.2
40 REM Author Keith McAlpine
50 REM RISC User March 1988
60 REM Program Subject to Copyright
70 :
80 ON ERROR GOTO 370
90 DIM reset$(15),redef$(15)
100 PROCinit
110 PROCdraw_screen
120 PROCselect(current%)
130 PROCupdate_screen
140 *POINTER 1_
150 :
160 REPEAT
170 PROCupdate_screen
180 REPEAT
190 MOUSE mx%,my%,button%
200 UNTIL button%<0
210 CASE TRUE OF
220 WHEN FNin_area(224,415,479,447):PR
OCchange(red%)

```

AT THE END OF THE RAINBOW



```

230 WHEN FNin_area(224,319,351,351):PR
OCchange(grn%)
240 WHEN FNin_area(224,223,479,255):PR
OCchange(blue%)
250 WHEN FNin_area(0,567,1279,951):PRO
Cnew colour
260 WHEN FNin_area(956,382,1152,486):P
ROCreset colour
270 WHEN FNin_area(956,254,1184,326):P
ROCshow vdus
280 WHEN FNin_area(956,126,1248,200):P
ROCprint vdus
290 WHEN FNin_area(956,30,1088,70):fin
ished%=TRUE
300 ENDCASE
310 UNTIL finished%
320 MODE 12
330 PRINT"The colour codes are:-"
340 PROCdisplay_vdus
350 END
360 :
370 MODE 12:ON ERROR OFF
380 REPORT:PRINT" at line ";ERL:END
390 :
400 DEFPROCinit
410 MODE 13:OFF
420 current%=0:finished%=FALSE
430 gcol%=0:tint%=0
440 FOR lp%=0 TO 15
450 SYS "OS_ReadPalette",lp%,16 TO r0,
r1,r2,r3
460 coll$=FNbinary(r2,32)
470 reset$(lp%)=coll$:redef$(lp%)=coll
$
480 NEXT lp%
490 ENDPROC
500 :

```

```

510 DEFPROCdraw screen
520 VDU23,224,255,129,129,129,129,129,
129,255
530 VDU23,225,255,255,255,255,255,255,
255,255
540 PRINTTAB(4,0)"256 Colour Mode Pale
tte Definer"
550 PRINTTAB(11,1)"By Keith McAlpine"
560 FOR y1%=0 TO 16 STEP 16
570 FORy2%=0TO1STEP1
580 FOR t%=0 TO 3 STEP 1
590 FOR x1%=0 TO 32 STEP 32
600 FOR x2%=0 TO 14 STEP 2
610 c%=x1%+x2%+y1%+y2%
620 x_position%=35*x1%/2+40*x2%
630 y_position%=927-(24*t%+12*y1%+96*y
2%)
640 GCOL c% TINT t%*64
650 RECTANGLE FILL x_position%,y_posit
ion%,80,24
660 NEXTx2%,x1%,t%,y2%,y1%
670 PRINTTAB(3,18)"RED"""" GREEN""""
BLUE"
680 GCOL 16 TINT 255
690 RECTANGLE FILL 956,382,196,104
700 RECTANGLE FILL 956,254,228,72
710 RECTANGLE FILL 956,126,292,72
720 RECTANGLE FILL 956,30,132,40
730 COLOUR 144 TINT 255
740 PRINTTAB(30,17)"RESET"TAB(30,18)"C
OLOUR"TAB(30,19);current%
750 PRINTTAB(30,22)"DISPLAY"TAB(30,23)
"COLOURs"
760 PRINTTAB(30,26)"PRINT OUT"TAB(30,2
7)"COLOURs"
770 PRINTTAB(30,30);"QUIT"
780 GCOL 15 TINT 255
790 RECTANGLE 956,382,196,104
800 RECTANGLE 956,254,228,72
810 RECTANGLE 956,126,292,72
820 RECTANGLE 956,30,132,40
830 COLOUR 128 TINT 0
840 ENDPROC
850 :
860 DEFFNbinary(num%,bytenum%)
870 bin$=""
880 REPEAT
890 bin$=STR$(num% MOD 2)+bin$
900 num%=num% DIV 2
910 UNTIL num%=0
920 bin$=STRING$(bytenum%-LEN(bin$),"0
"+bin$
930 =bin$
940 :

```

AT THE END OF THE RAINBOW

```

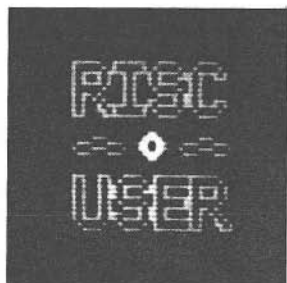
950 DEFPROCupdate_screen
960 COLOUR 15 TINT 255
970 PRINTTAB(9,15)"PALETTE ";current%;
",",red%,"",grn%,"",blu%;SPC6
980 PRINTTAB(9,16)"GCOL ";gcol%," TINT
";tint%;SPC3
990 COLOUR 3 TINT 255
1000 PRINTTAB(8,18);STRING$(red%>>4,CHR
$225);STRING$(7-(red%>>4),CHR$224)
1010 COLOUR 12 TINT 255
1020 PRINTTAB(8,21);STRING$(grn%>>4,CHR
$225);STRING$(3-(grn%>>4),CHR$224)
1030 COLOUR 48 TINT 255
1040 PRINTTAB(8,24);STRING$(blu%>>4,CHR
$225);STRING$(7-(blu%>>4),CHR$224)
1050 COLOUR current%,red%,grn%,blu%
1060 PROCdecode(redef$(current%),red%,g
rn%,blu%)
1070 ENDPROC
1080 :
1090 DEFPROCcode(str$,RETURN r$,RETURN
g$,RETURN b%)
1100 r$=MID$(str$,17,8):g$=MID$(str$,9,
8)
1110 b$=LEFT$(str$,8):r%=EVAL("%"+r$)
1120 g%=EVAL("%"+g$):b%=EVAL("%"+b$)
1130 ENDPROC
1140 :
1150 DEFPROCdecode(RETURN str$,r%,g%,b%
)
1160 r$=FNbinary(r%,8):g$=FNbinary(g%,8
)
1170 b$=FNbinary(b%,8):str$=b$+g$+r$+"0
0010000"
1180 ENDPROC
1190
1200 DEFPROCselect(current%)
1210 PROCcode(redef$(current%),red%,grn
%,blu%)
1220 COLOUR 144 TINT 255
1230 COLOUR 63 TINT 255
1240 PRINTTAB(30,19);current%;SPC2
1250 COLOUR 128 TINT 0
1260 ENDPROC
1270
1280 DEFNin_area(lx%,by%,rx%,ty%)
1290 IF mx%>=lx% AND mx%<=rx% AND my%>=
by% AND my%<=ty% THEN =-1
1300 =0
1310 :
1320 DEFPROCchange(RETURN col%)
1330 IF mx%<=256 THEN col%=0:ENDPROC
1340 col%=((mx%-224) DIV 32)*16)
1350 ENDPROC
1360
1370 DEFPROCnew_colour
1380 current%=(951-my%)DIV24
1390 gcol%=POINT(mx%,my%)
1400 tint%=TINT(mx%,my%)
1410 PROCselect(current%)
1420 ENDPROC
1430 :
1440 DEFPROCreset_colour
1450 redef$(current%)=reset$(current%)
1460 PROCcode(redef$(current%),red%,grn
%,blu%)
1470 ENDPROC
1480 :
1490 DEFPROCshow_vdus
1500 COLOUR 63 TINT 255
1510 PRINTTAB(0,18)"Press Shift to scro
ll."
1520 COLOUR 15 TINT 255
1530 VDU 28,0,31,28,20,12,14
1540 PROCdisplay_vdus
1550 VDU 26:COLOUR 63 TINT 255
1560 PRINTTAB(0,18)"Press any key to co
ntinue."
1570 key=GET
1580 VDU 28,0,31,28,18,12,13,26
1590 PRINTTAB(3,18)"RED"""" GREEN""""
BLUE"
1600 PROCupdate_screen
1610 ENDPROC
1620 :
1630 DEFPROCdisplay_vdus
1640 FOR lp%=0 TO 15
1650 PROCcode(redef$(lp%),r%,g%,b%)
1660 PRINT"COLOUR ";lp%;"",r%;"",g%;"",
",b%
1670 NEXT lp%
1680 ENDPROC
1690 :
1700 DEFPROCprint_vdus
1710 LOCAL ERROR
1720 ON ERROR LOCAL OFF:GOTO1800
1730 COLOUR 63 TINT 255
1740 PRINTTAB(0,23)"Press Escape to abo
rt."
1750 VDU 28,0,31,28,24,12,14
1760 COLOUR 15 TINT 255
1770 VDU 2
1780 PRINT"256 colour palette COLOUR de
finitions"
1790 PROCdisplay_vdus
1800 VDU 3,28,0,31,28,23,12,13,26
1810 RESTORE ERROR
1820 ENDPROC

```

A MOUSE POINTER DESIGNER

by Mark Davis

Design your own stylised pointers with this short, easy-to-use program.



The Archimedes firmware provides a single standard mouse pointer activated with "Pointer. The operating system permits up to 4 resident pointer definitions of 32 by 32 pixels. The present program makes use of this fact, and allows the creation of user-defined pointers. Perhaps the most important feature of the program is that the data statements for defining the pointer are laid out in such a way that you can easily design your own pointer. At present, the program creates a "RISC USER" pointer as pointer number 2 (line 290), but by using an editor in "over-write" mode you can easily alter the program to suit your own requirements.

The data grid in lines 300-620 is set out as follows: transparent parts of the pointer are represented by dots, while a 1, 2 or / indicate a pixel in one of 3 colours. These are defined (in red green and blue components) in lines 640-660. The only other thing to note is the position of the active part of the pointer (i.e. the part which "points") - in our example it is dead centre. The active point is defined by the position of the two asterisks. The one on the first data line indicates the pixel column of the active point, while the one at the end of the data line (at line 460) indicates the active pixel row. A little experiment should clarify how it all works.

```

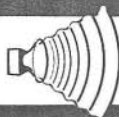
10 REM >Pointer5
20 REM Program User defined pointer
30 REM Version A 0.1
40 REM Author Mark J Davis
50 REM RISC User March 1988
60 REM Program Subject to Copyright
70 :
80 DIM block 10,data 256
90 MODE9:READ number
100 READ A$:xpos=INSTR(A$,"*"):ypos=0
110 FOR X%=0 TO 31:P%=data+X%*8
120 READ A$:FOR Y%=0 TO 7:Q%=0
130 FOR Z%=0 TO 7 STEP 2

```

```

140 N%= (ASC MID$(A$,Y%*4+Z%/2+1,1)-46)<<Z%
150 Q%=Q%+N%:NEXT Z%:P%=P%+Q%:NEXT Y%
160 IF RIGHT$(A$,1)="/" THEN ypos=X%
170 NEXT X%
180 ?block=0:block?1=number:block?2=8
190 block?3=&20:block?4=xpos
200 block?5=ypos:block?6=data
210 SYS 7,&15,block
220 *POINTER
230 FOR C%=1 TO 3:READ R%,G%,B%
240 MOUSE COLOUR C%,R%,G%,B%
250 NEXT
260 MOUSE ON 2
270 END
280 :
290 DATA 2 :REM Pointer no.
300 DATA .....*.....
310 DATA ////////////////
320 DATA / ./ ./ ./ ./ ./
330 DATA / /// /// /// /// ///
340 DATA / ./ ./ ./ ./ ./
350 DATA / /// ./ ./ ./ ./
360 DATA / ./ ./ ./ ./
370 DATA / // ./ // /// //
380 DATA / ./ ./ // /// //
390 DATA / ./ /// /// ///
400 DATA / ./ // ./ //
410 DATA ////////////////
420 DATA .....
430 DATA .....
440 DATA .....
450 DATA ...../11/.....
460 DATA /// //.../1..1/...// *
470 DATA / /// //1..1/ //
480 DATA //...//.../11/...//
490 DATA .....//.....
500 DATA .....
510 DATA .....
520 DATA //...//...//.....
530 DATA / ./ ./ ./
540 DATA / ./ // /// //
550 DATA / ./ // //...// ./
560 DATA / ./ // /// //
570 DATA / ./ ./ //
580 DATA / ./ // /// //
590 DATA / ./ //...// //
600 DATA / /// // /// //
610 DATA / ./ // //
620 DATA ..//...//...//...//
630 REM Colour Definitions
640 DATA 64,190,220 :REM Colour 0
650 DATA 220,0,64 :REM Colour 1
660 DATA 240,240,240 :REM Colour 1

```



THE RISC USER VOICE GENERATOR (continued from page 7)

```

1560 LDRB R1,[R6,R1,LSL#1]
1570 MOV R1,R1,LSR#1:RSB R1,R1,#127
1580 .fillloop
1590 ADD R2,R2,R2,LSL#16
1600 LDRB R0,[R5,R2,LSR#24]
1610 SUBS R0,R0,R1,LSL#1
1620 MOVMI R0,#0:STRB R0,[R12],R11
1630 ADD R2,R2,R2,LSL#16
1640 LDRB R0,[R5,R2,LSR#24]
1650 SUBS R0,R0,R1,LSL#1
1660 MOVMI R0,#0:STRB R0,[R12],R11
1670 ADD R2,R2,R2,LSL#16
1680 LDRB R0,[R5,R2,LSR#24]
1690 SUBS R0,R0,R1,LSL#1
1700 MOVMI R0,#0:STRB R0,[R12],R11
1710 ADD R2,R2,R2,LSL#16
1720 LDRB R0,[R5,R2,LSR#24]
1730 SUBS R0,R0,R1,LSL#1
1740 MOVMI R0,#0:STRB R0,[R12],R11
1750 CMP R12,R10:BLT fillloop
1760 SUBS R4,R4,#1:LDR R1,[R9]
1770 STMIA R9,{R1-R8}
1780 MOVPL R0,#%00001000
1790 MOVMI R0,#%00000010
1800 LDMFD R13!,{PC}
1810 .gateoff MOV R0,#0
1820 .floop
1830 STRB R0,[R12],R11
1840 STRB R0,[R12],R11
1850 STRB R0,[R12],R11
1860 STRB R0,[R12],R11
1870 CMP R12,R10:BLT flop
1880 MOV R0,#%00000001
1890 LDMFD R13!,{PC}
1900 .voicename EQU$ name$
1910 EQU$B:ALIGN
1920 ]
1930 NEXT PASS:ENDPROC
1940 :
1950 DEFPROCRC:FORX=1TO32
1960 SYS "Sound_RemoveVoice",0,X:NEXT
1970 ENDPROC
1980 DEFPROCfill(A$,name$):addr=voice%:
PROCwavass(name$,voice%):voice%+=1
1990 PRINT"Assembling Voice ",addr
2000 addr=wavetable%+addr*256
2010 N=FNblock(1):IFINSTR(A$,"RND")>0 T
HENENDPROC
2020 Q=FNblock(N):ENDPROC
2030 :
2040 DEFFNblock(M):N=0
2050 FORX=0TO255:Z=RAD(X*360/256):Q=EVA
L(A$)*M

2060 SYS "Sound_SoundLog",&FFFFFF*Q TO
N%
2070 SYS "Sound_LogScale",N% TO X?addr
2080 IF(ABSQ) >N THENN=ABSQ
2090 NEXTX:=127/N
2100 :
2110 DEFPROCaenv(step,number)
2120 E%!aenvtable%=(ABSstep AND&FF)-&10
0*(step<0)+(number AND&7FFF)*&10000
2130 E%+=4:ENDPROC
2140 :
2150 DEFPROCfenv(step,number)
2160 F%!fenvtable%=(ABSstep AND&FF)-&10
0*(step<0)+(number AND&7FFF)*&10000
2170 F%+=4:ENDPROC

* * * * *

10 REM >Effects4
20 PROCclear
30 *RMLoad UserVoices
40 PROCchannels
50 PROCkeys
60 PROCplay
70 END
80 :
90 DEFPROCplay
100 REPEAT
110 FOR N=50 TO 200 STEP 20
120 FOR C=7 TO 1 STEP -1
130 SOUND C,-15,N,50
140 X=INKEY(30):NEXT:NEXT:UNTIL FALSE
150 ENDPROC
160 :
170 DEFPROCclear
180 FOR X=1 TO 32
190 SYS "Sound_RemoveVoice",0,X:NEXT
200 ENDPROC
210 :
220 DEFPROCchannels
230 FOR A=1 TO 7
240 OSCLI("CHANNELVOICE "+STR$(A)+" "+
STR$(A))
250 NEXT:VOICES 8:*VOICES
260 ENDPROC
270 :
280 DEFPROCkeys
290 FOR A=1 TO 7
300 OSCLI("KEY"+STR$(A)+" SOUND "+STR$
(A)+"",-15,100,50|M")
310 NEXT
320 ENDPROC

```

THE ARCHIMEDES I/O PODULE

Reviewed by Lee Calcraft

Acorn's newly-released I/O Podule brings further BBC micro compatibility to Archimedes users. Yet its provision of analogue port, digital user port, 1 MHz bus, and optional MIDI interface will appeal to Beeb and non-Beeb users alike. The podule consists of a single PCB of around 170 x 180 mm in size. This is twice the width of podules such as the CC ROM board reviewed last month, and will take up the space of two podules of the possible 4 on a 400 series machine. On a 300 series machine, which will only take two podules at maximum, it still leaves room for a second. This could conceivably be a second I/O podule, though I am not aware that this possibility has been explored.

The I/O podule plugs into a socket on the podule backplane. The backplane itself is an optional extra on 300 series machines, though it comes as standard with the 400 series. Fitting the I/O Podule is extremely easy, since all the interface sockets on the board are already fitted to the back panel, which simply replaces one of the blank panels at the rear of the machine.

Analogue Port

4 analogue input channels
8 bit accuracy
Conversion rate 10 msec/channel
Two "fire" button lines

User Port

Single 8-bit bi-directional port
with timers and control lines

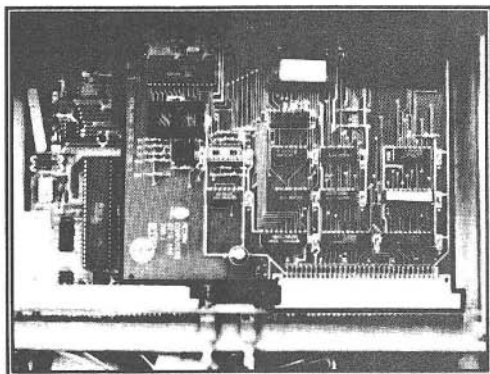
1 MHz bus

Read/write access to two 256-byte memory-mapped blocks at 1MHz clock rate

Features of the I/O Podule Interface

Once installed, you should check that the resident module called **Podule** is not software-unplugged. Then at power-up the podule module (!) will automatically load into the RMA one or two further modules called **I/O_Podule** and **MIDI** (both resident in EPROM on the I/O Podule board). MIDI is only loaded if your

board has the MIDI upgrade. For the purposes of this review we will assume that the MIDI module is not present, though users with this option should note that when the MIDI module is installed, it sets the number of voices on the sound system to 8 (the default is 1). This dramatically reduces sound output (VDU7 is barely audible). Acorn have agreed to fix this in future versions of the software, but for the moment, you may wish to use *UNPLUG MIDI to remove the module until it is required.



View of I/O Podule installed on an A440

The I/O podule relocatable module contains all the software for implementing the I/O board interfaces, and essentially provides a set of operating system calls to send data to and from each port. The majority of these are implemented as OSBYTE calls, similar to those used on the Model B and Master. In a similar way to earlier BBC micros, the interfaces on the podule are address-mapped, and the areas of memory used are given the same names: Jim and Fred for the 1MHz bus, and Sheila for the user and analogue ports. Of course Jim, Fred and Sheila are all controlled by Arthur! - and the base address is no longer &FE00, as it is on the BBC micro, but is dependent on which of the 4 possible podule sockets are used.

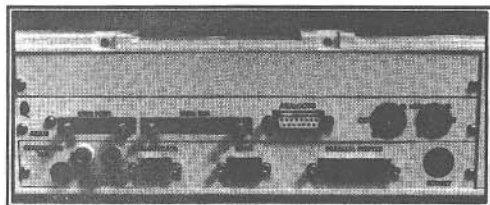
Acorn have implemented a special call which returns the I/O podule base address, but unfortunately both the call address, and the call name are wrongly given in the manual

THE ARCHIMEDES I/O PODULE

(actually, they are not even close!). The correct call number is SYS &40500 or SYS "I/O_Podule_Hardware". Thus if you type:

```
SYS &40500 TO A,B  
PRINT~B
```

you will get the required base address. With the I/O board in socket zero (use *Podules to find out), I obtained the result: &33C0000. Unfortunately, all attempts to access this area (or the area at &33C2000 given in the manual) gave "Abort on data transfer" errors, so there may be more errors in the manual. Generally speaking of course, the user should not need to know the base address of his podule hardware, since he can fully access all ports with the system calls provided.



Rear of A440 showing I/O connectors

TESTING OUT THE PODULE

An easy way to check that your I/O podule is functioning correctly is to type:

```
PRINT ADVAL 2
```

With no device plugged into channel 2 of the analogue port you will get a result of around 53700. If Arthur can't find Sheila then you will get a "Bad command" message. As a test of the analogue port, I checked it out with a pair of BBC micro joysticks. These worked quite satisfactorily, though if you are going to use the port for precise voltage measurements, you will be disappointed to hear that no accurate reference source is supplied. The analogue to digital converter still makes use of the same reference source as the old Beeb, even though to upgrade this would have been a trivial matter.

When testing the user port, I found the manual quite misleading. It implies that the ARM equivalent of OSBYTE 150 and 151 can be used to read and write to the 16 registers of the 6522 VIA used by the podule. In fact the podule software will not let you access registers 1 and 3 (at Sheila offset &61 and &63). These two registers give access to port A of the VIA, and this is used for other purposes, including ROM paging.

To write the value 128 to register 0 of the VIA, use:

```
*FX151, 96, 128 (96=&60+0)
```

To read register 4 of the VIA, use:

```
SYS 6, 150, &64 TO A,B,C  
PRINT C
```

Examples of accessing the VIA

Thus a test program which I was using to read and display all VIA registers repeatedly crashed. But once I stopped it from reading the two prohibited registers it worked ok. This is not fully compatible with the old Beeb, where no block is placed on the two port A registers. One other small point of incompatibility is that the VIA on the podule runs twice as fast as that of the Beeb. But the new 2MHz clocking speed will only really affect the VIA's timers, which will run at twice the speed of those on a Beeb.

In conclusion, Acorn have produced a creditable I/O podule whose strength lies in its high degree of compatibility with the Beeb's hardware. The many users of the Beeb's excellent set of interfaces, including those in schools and colleges, who have upgraded to the Archimedes should find this an invaluable product.

RU

The Acorn I/O Podule costs £90.85
The MIDI add-on to this podule costs £33.35
Prices include VAT.
Acorn are on (0223) 214411

ARCHIMEDES

ROM/RAM PODULE

This podule accepts a variety of ROM or RAM chips and includes ROM and RAM filing system software.

Seven 32 pin sockets can accept any combination of ROMs (16K to 128K) or RAMs (8K to 128K)

Battery and controller chip. Trickle charged battery ensures RAM contents are retained whilst computer is off (optional).

The controller ROM provided contains ROM and RAM filing system software, is fast, and totally Arthur compatible.

Links for battery backup or normal supply

Special controller circuitry automatically detects the type and size of chip plugged into each socket—eliminating 32 links

Edge connector plugs into podule back-plane of 300 or 400 series Archimedes

The board accepts software such as the Archimedes versions of Inter-Word, Inter-Chart and Inter-Sheet and will accept the forthcoming document processor and drawing programs. It is compatible and fully obeys all Acorn podule standards. It requires the back-plane to be installed—available from us and Acorn stockists.

£49.00 + VAT (£56.35) standard
£59.00 + VAT (£67.85) with battery
and controller chip



Computer Concepts Ltd

Gaddesden Place

Hemel Hempstead

Herts HP2 6EX.

Telephone 0442 63933

ACCESS & BARCLAYCARD ACCEPTED

HINTS & TIPS HINTS & TIPS

Another crop of Hints and Tips, rounded up by Lee Calcraff.

FULL WIPE, FULL COPY

On OS 1.2 the command:

*WIPE * R F ~C

will wipe the contents of a whole disc, including all directories, without prompting of any kind. This can sometimes be quicker than reformatting a disc to clear it. But take care not to issue the command unless you really mean it - recovering files deleted in this way is possible, but it can be hard work.

Again on OS 1.2, the command:

*COPY :0 :1 Q ~V ~P ~C R

will make a complete copy on drive 1 of the disc in drive 0, creating new directories where necessary. It has 3 possible advantages over *BACKUP. It can be much quicker (providing there are not too many files), it will work between different sized discs (*BACKUP will not), and it will not erase any files or directories already held on the destination disc.

DOUBLE VISION IN MODE 7

Have you ever wondered why mode 7 takes up a massive 80K of RAM, when mode 9 for example, gets by with only 40K? The answer is that Arthur keeps two copies of the screen in this mode in order to produce flashing colours. Every second or so, the operating system switches between the two banks to create the flashing effect. Thanks to David Pilling for this information.

800K FORMAT FOR SPEED

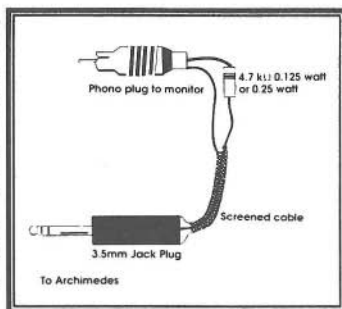
To save time in formatting discs, you might be tempted on occasion to use the 600K format. Be warned that all save and load operations on 600K format discs take many times longer than on 800K discs (see the timings in this month's A440 review). The reason for the excellent performance of the 800K size is that disc sectors are exactly 1K in length.

AMPLIFIED AUDIO

To greatly improve the volume and quality of the Archimedes sound output, it is possible to connect a lead from the headphone output at the back of the machine to the input of the audio amplifier at the rear of the colour monitor. For this you will need a phono plug for the monitor, and a 3.5 mm stereo jack plug. Either of the two stereo output channels may be used, with a 4.7k resistor in series with the lead. The earth leads should be directly

connected.

Once the lead is installed, you can adjust the output level with the volume control on the front panel of the monitor. Thanks to Rob Barnes for this one.



DIRECTORIES LAST

To ensure that directories are grouped together in displays of the disc catalogue (either from *CAT or from within the Desktop or the RISC User disc menu), precede the names of all directories with the character formed by Shift £. Thanks to Barry Christie for this hint.

32-BIT ERRORS

On the model B, error numbers are in the range 0 to 255. On the Archimedes, the error number is a 32-bit word. On this system, the bottom byte of the error number gives the error type, and is compatible with BBC micro error numbers. To achieve full compatibility, you should therefore use ERR AND &FF. The top three bytes of the 32-bit error number may be used to identify which part of the system firmware caused the error (see the Reference Manual part 1 page 10). For Basic errors (though not filing system errors), the top three bytes are always zero, giving complete compatibility with the model B. Thanks to Rob Pickering for this hint.

EASIER OSWORD

OSWORD calls are easily made on the series one operating system. The syntax is:

SYS "OS_Word", number, address

where number is the OSWORD number, and address is the start address of the parameter block used for passing data to and from the call. There is no need for all that DIV and MOD business familiar to Model B and Master users.

OSBYTE ERRORS

In the operating system test given in last month's Hints, we used a special feature of OSBYTE calls in which, if bit 17 of the call number is set, any errors generated will

HINTS & TIPS

cause the error number and the error message to be placed in memory, and the call return with R0 holding the start address of this information. The first four bytes of this memory block contain the error number. This is immediately followed by an ASCII string giving the error message, which is itself terminated by a zero byte. By calling OSBYTE zero with zero in R1 we caused an error to be generated, and the resultant string is thereby stored.

On the series one operating system, OSBYTE can be called using SYS "OS_Byte" in place of SYS 6 (see Hints and Tips, Issue 2). The equivalent to SYS 6+2^17 used above, is the new call SYS "XOS_Byte".

CALLING TWIN

Twin is normally called from Basic with the keyword TWIN. If you call it with:

TWINO n

where n lies between 0 and 15, TWIN will be entered with any resident Basic program displayed as if it had been listed under LISTO n. A particularly useful option is TWINO 8, because this strips off the line numbers. When you return to Basic the program is then automatically

HINTS & TIPS

renumbered. This makes moving chunks of code around within a program a very easy matter indeed.

As you may know, TWIN is (unfortunately) not supplied as a relocatable module, but as a piece of absolute code. My version runs at &40000. You can make use of its start address to avoid loading it in from disc every time. If you type:

CALL &40000

you will engage Twin - providing of course that you have previously installed it (using TWIN from Basic or *TWIN).

KEYBOARDLESS BOOT

If the Archimedes is being used for display purposes, the keyboard can be disconnected. When the machine is switched on, the system will automatically execute the IBOOT file in the default drive. Thanks to Rob Barnes for this hint.

RU

Please send your Hints & Tips to the Editors at the editorial address given at the end of the magazine. All contributions welcomed.

Archimedes

5.25 Disk Interface.

Fit a 5.25 Drive to your Machine the safe and easy way using our
NEW DISK INTERFACE

- ◆ Available Now
- ◆ No Soldering
- ◆ Supports 4 Drives
- ◆ Low Power Consumption
- ◆ Fully Buffered
- ◆ Easy to Fit
- ◆ All Cables Supplied
- ◆ Full Instructions

In Stock Now and Despatched the same day on receipt of your
Cheque / Postal Order for £24.00 inc P&P.

Dudley Micro Services

30 Hadley Close, Netherton, Dudley, West Mids. DY2 9JX

Tel: (0384) 633142

RISC USER magazine

MEMBERSHIP

RISC User is available only on subscription, with a special introductory rate until early 1988. Full subscribers to RISC User may also take out a reduced rate subscription to BEEBUG (the magazine for the BBC micro and Master series).

All subscriptions, including overseas, should be in pounds sterling. We will also accept payment by Connect, Access and Visa, and official UK orders are welcome.

RISC USER SUBSCRIPTION RATES

£14.50	1 year (10 issues) UK, BFPO, Ch.I
£20.00	Rest of Europe & Eire
£25.00	Middle East
£27.00	Americas & Africa
£29.00	Elsewhere

RISC USER & BEEBUG

£23.00
£33.00
£40.00
£44.00
£48.00

BACK ISSUES

We intend to maintain stocks of back issues. New subscribers can therefore obtain earlier copies to provide a complete set from Vol.1 Issue 1. Back issues cost £1.20 each. You should also include postage as shown:

Destination	UK, BFPO, Ch.Is	Europe plus Eire	Elsewhere
First Issue	40p	75p	£2
Each subsequent Issue	20p	45p	85p

MAGAZINE DISC

The programs from each issue of RISC User are available on a monthly 3.5" disc. This will be available to order, or you may take out a subscription to ensure that the disc arrives at the same time as the magazine. The first issue (with six programs and animated graphics demo) is at the special low price of £3.75. The disc for each issue contains all the programs from the magazine, together with a number of additional items by way of demonstration, all at the standard rate of £4.75.

MAGAZINE DISC PRICES

	UK	Overseas
Single issue discs	£ 4.75	£ 4.75
Six months subscription	£25.50	£30.00
Twelve months subscription	£50.00	£56.00

Disc subscriptions include postage, but you should add 50p per disc for individual orders.

All orders, subscriptions and other correspondence should be addressed to:

RISC User, Dolphin Place, Holywell Hill, St Albans, Herts AL1 1EX.
Telephone: St Albans (0727) 40303
(24hrs answerphone service for payment by Connect, Access or Visa card)